

ENKBD-2.0

VAX 11/730 8085
TEST PART IV

EP-ENKBD-DL-2.0
FICHE 1 OF 1

JAN 1983
COPYRIGHT © 82-83
MADE IN USA



IDENTIFICATION

```
Product code:  ENKBD VERSION 2.0
Product name:   8085 BASED MICRO-DIAGNOSTIC FOR VAX-11/730 WCS
                AND CPU MODULES
Product date:   11-OCT-82
Maintainer:     BASE SYSTEMS DIAGNOSTIC ENGINEERING
```

The information in this document is subject to change without notice and should not be construed as a commitment by Digital Equipment Corporation. Digital Equipment Corporation assumes no responsibility for any errors that may appear in this document.

The software described in this document is furnished under a license and may be used or copied only in accordance with the terms of such license.

No responsibility is assumed for the use or reliability of software on equipment that is not supplied by Digital or its affiliated companies.

Copyright (c) 1982 by Digital Equipment Corporation. All Rights Reserved.

The following are trademarks of Digital Equipment Corporation.

DEC
DECUS
UNIBUS

DECsystem-10
MASSBUS
VAX

DECSYSTEM-20
PDP
VMS

digital

Table of Contents

1.0	ABSTRACT	3
2.0	HARDWARE REQUIREMENTS	3
3.0	SOFTWARE REQUIREMENTS	3
4.0	PREREQUISITES	3
5.0	OPERATING INSTRUCTIONS	3
5.1	Options	3
5.2	Event Flags	4
5.3	APT Setup	4
6.0	PROGRAM FUNCTIONAL DESCRIPTION	4
6.1	Program Overview	4
6.2	Program Size	4
6.3	Program Run Times	4
6.4	Run-time Dynamics	4
6.5	Fault Detection	5
6.6	Performance During Hardware Failures	5
6.7	Program Applications	5
6.8	Test Descriptions	6
7.0	MAINTENANCE HISTORY	6

Not applicable

[illegible]

Not applicable

5.3 APT Setup

Under APT, the diagnostic will halt on error and report error information to APT via the mailbox in 8085 RAM.

The following describes the APT parameters needed to execute this diagnostic:

6.0 PROGRAM FUNCTIONAL DESCRIPTION

6.1 Program Overview

This program is designed to detect stuck-at faults and provide a repair tool which helps isolate the failing component. A bottom up diagnostic strategy is utilized which builds on previous tests. The tests should be run in consecutive order, to gain the best isolation of a failing component.

6.2 Program Size

This program runs in 8085 RAM memory and is approximately 3K bytes long.

6.3 Program Run Times

This program will take approximately 10 seconds to run.

6.4 Run-time Dynamics

Not applicable

[illegible]

Customers may use the program to verify the basic integrity of the WCS and CPU modules in the system.

[illegible]

See the actual test in the listing for detailed descriptions and error analyses. A brief description of the logic being tested by each test can be found in the table of contents of the listing.

DATE	VERSION	DESCRIPTION
8-MAR-82	1.0	Initial release.
11-OCT-82	2.0	RD changes.

ENKBD 8085 based diagnostic for WCS/DAP module Rev 02.00
Table of Contents

7000	3	:	134	DATA AREA
7000	4	:	323	PROGRAM ENTRY POINT
7000	5	:	342	VARIABLES AREA
7000	6	:	397	PROGRAM SPECIFIC SUBROUTINES
7000	7	:	398	CHECK RD
7000	8	:	417	MASTER_RESET
7000	9	:	432	MASTER_RESET_RESTORE
7000	10	:	446	CLR_CG_PNT
7000	11	:	460	INC_CG_PNT
7000	12	:	474	WRITE_MASTER
7000	13	:	491	READ_MASTER
7000	14	:	510	WRITE_MODE
7000	15	:	530	READ_MODE
7000	16	:	550	WRITE_LOAD
7000	17	:	569	READ_LOAD
7000	18	:	589	WRITE_HOLD
7000	19	:	609	READ_HOLD
7000	20	:	628	WRITE_ALARM
7000	21	:	653	READ_ALARM
7000	22	:	678	WRITE_ITDB
7000	23	:	698	READ_ITDB
7000	24	:	716	MM_BITS
7000	25	:	737	INC_CYCLE_CNT
7000	26	:	749	DISARM_CNTR
7000	27	:	777	LOAD_CNTR
7000	28	:	805	SAVE_CNTR
7000	29	:	832	SET_OUTPUT
7000	30	:	850	CLR_OUTPUT
7000	31	:	865	INC_CNTR
7000	32	:	882	LOAD_MOD_DAT
7000	33	:	897	TESTE_ERROR
7000	34	:	927	TEST1F_ERROR
7000	35	:	960	TEST21_ERROR
7000	36	:	993	TEST22_ERROR
7000	37	:	1023	TEST24_ERROR
7000	38	:	1052	TEST26_ERROR
7000	39	:	1087	TEST20_ONES_ZEROS
7000	40	:	1047	INTERVAL_TIMER TESTS
7000	41	:	1230	TEST 1E - MASTER RESET TEST
7000	42	:	1397	TEST 1F - MASTER MODE REGISTER TEST
7000	43	:	1571	TEST 20 - ELEMENT CYCLE REGISTER TEST
7000	44	:	1790	TEST 21 - ALARM REGISTER TEST
7000	45	:	1985	TEST 22 - DATA POINTER SEQUENCING TEST
7000	46	:	2279	TEST 23 - COUNTER LOAD AND SAVE TEST
7000	47	:	2439	TEST 24 - OUTPUT BIT SET/CLEAR TEST
7000	48	:	2590	TEST 25 - COUNTER CARRY TEST
7000	49	:	2704	TEST 26 - COMPARATOR TEST
7000	50	:		
7000	51	:		
7000	52	:		
7000	53	:		
7000	54	:		
7000	55	:		
7000	56	:		
7000	57	:		
7000	58	:		
7000	59	:		
7000	60	:		
7000	61	:		
7000	62	:		

7000 63
7000 64
7000 65
7000 66
7000 67
7000 68
7000 69
7000 70
7000 71
7000 72
7000 73
7000 74
7000 75
7000 76
7000 77
7000 78
7000 79
7000 80
7000 81
7000 82
7000 83
7000 84
7000 85
7000 86
7000 87
7000 88
7000 89
7000 90
7000 91
7000 92
7000 93
7000 94
7000 95
7000 96
7000 97
7000 98
7000 104
7000 105
7000 106
7000 107
7000 108
7000 109
7000 110
7000 111
7000 112
7000 113
7000 114
7000 115
7000 116
7000 117
7000 118
7000 119
7000 120
7000 121
7000 122
7000 123
7000 124
7000 125
7000 126
7000 127

.TITLE 'ENKBD 8085 based diagnostic for WCS/DAP module Rev 02.00'

.....
COPYRIGHT (c) 1982 BY
DIGITAL EQUIPMENT CORPORATION, MAYNARD,
MASSACHUSETTS. ALL RIGHTS RESERVED.

THIS SOFTWARE IS FURNISHED UNDER A LICENSE AND MAY BE USED AND COPIED
ONLY IN ACCORDANCE WITH THE TERMS OF SUCH LICENSE AND WITH THE INCLUSION
OF THE ABOVE COPYRIGHT NOTICE. THIS SOFTWARE OR ANY OTHER COPIES THEREOF
MAY NOT BE PROVIDED OR OTHERWISE MADE AVAILABLE TO ANY OTHER PERSON. NO
TITLE TO AND OWNERSHIP OF THE SOFTWARE IS HEREBY TRANSFERRED.

THE INFORMATION IN THIS SOFTWARE IS SUBJECT TO CHANGE WITHOUT NOTICE, AND
SHOULD NOT BE CONSTRUED AS A COMMITMENT BY DIGITAL EQUIPMENT CORPORATION.

DIGITAL ASSUMES NO RESPONSIBILITY FOR THE USE OR RELIABILITY OF ITS
SOFTWARE ON EQUIPMENT THAT IS NOT SUPPLIED BY DIGITAL.

..++

FACILITY: VAX-11/730 MICRO-DIAGNOSTIC.

ABSTRACT: This 8085 based diagnostic tests the hardcore logic of the
WCS and DAP modules.

ENVIRONMENT: ENKAA Micro-diagnostic Monitor

AUTHOR: Jim Klumpp 4-MAY-81

MODIFIED BY: David Mayo 8-MAR-82 VERSION 01.00
Ernie Preisig 2-SEP-82 VERSION 02.00 RD changes

..--


```

7000 128
7000 129
7000 130
7000 131
7000 132
7000 133
7000 134 000
7000 135
7000 136
7000 137
7000 138
7000 139
7000 140
7000 141
7000 142
7000 143 00000000
7000 144
7000 145 00000000
00000002
00000004
00000006
00000006
00000001
00000003
00000005
00000007 00000006
7000 146
7000 147 00000000
7000 148
7000 149
7000 150 00000001
7000 151 00000001
7000 152
7000 153
7000 154 00000002
7000 155
7000 156 00000002
7000 157
7000 158
7000 159 00000003
7000 160
7000 161 00000003
7000 162
7000 163
7000 164 00000004
7000 165 00000004
7000 166
7000 167 00000005
7000 168
7000 169 00000006
7000 170
7000 171 00000007
7000 172
7000 173 00000007
7000 174
7000 175 00000008
7000 176 00000009
7000 177 0000000A
7000 178
7000 179 0000001C
    
```

```

.PSECT CPU4,BYTE
.ALIGN BYTE
    
```

```

:*****
: DATA AREA
:*****
    
```

```

HEAD =0
    
```

```

EQUATE
    
```

```

;REGISTER EQUATE MACRO
    
```

```

UPC14A      =<^X00>      ;(RO) UPC BIT 14 ASSERTED HIGH <MSB>

MISC2       =<^X01>
BOOTF       =<^X01>      ;(RO) BOOT FLAG. ASSERTED LOW. <LSB>

DCLO        =<^X02>      ;(RO) DC LOW FLAG ASSERTED HIGH<MSB>
HLTFLG      =<^X02>      ;(RO) HALT FLAG ASSERTED LOW <LSB>

READJ1      =<^X03>      ;(RO) JUMPER J1 FLAG ASS. HIGH <MSB>
ALLOWP      =<^X03>      ;(RO) ENABLE CONTROL P ASS LOW <LSB>

READJ2      =<^X04>      ;(RO) BIT 07 EQUALS J2 <MSB>
APTFLG      =<^X04>      ;(RO) APT PRESENT FLAG <LSB>

AUTO        =<^X05>      ;(RO) AUTO TEST FLAG ASS LOW <LSB>

REMDIS      =<^X06>      ;(RO) REMOTE DISABLE FLAG ASS LOW <LSB>

OKV15       =<^X07>      ;(RO) 15 VOLTS OK FLAG ASS LOW <MSB>
BOOTEN      =<^X07>      ;(RO) BOOT ENABLE FLAG ASS LOW <LSB>

WCSB0       =<^X08>      ;(WO) WCS WRITE DATA REGISTER
WCSB1       =<^X09>
WCSB2       =<^X0A>

ITDB        =<^X1C>      ;(R/W) TIMER DATA REGISTER
    
```


7000 180 0000001D	ITCSR	=<^X1D>	;(R/W) TIMER STATUS REGISTER
7000 181			
7000 182 00000020	CLRCLK	=<^X20>	;(WO) CLEAR CPU CLOCK RUN
7000 183 00000021	SETCLK	=<^X21>	;(WO) SET CPU CLOCK RUN
7000 184			
7000 185 00000022	CLRSS	=<^X22>	;(WO) CLEAR CLOCK SINGLE STEP
7000 186 00000023	SETSS	=<^X23>	;(WO) SET CLOCK SINGLE STEP
7000 187			
7000 188 00000024	CLRCSK	=<^X24>	;(WO) CLEAR CSR CLOCK
7000 189 00000025	SETCSK	=<^X25>	;(WO) SET CSR CLOCK
7000 190			
7000 191 00000026	CLRUPK	=<^X26>	;(WO) CLEAR THE UPC CLOCK
7000 192 00000027	SETUPK	=<^X27>	;(WO) SET THE UPC CLOCK
7000 193			
7000 194 00000028	SETMCI	=<^X28>	;(WO) SET MEMORY CONTROL INIT
7000 195 00000029	CLRMCI	=<^X29>	;(WO) CLEAR MEMORY CONTROL INIT
7000 196			
7000 197 0000002A	SETMCK	=<^X2A>	;(WO) SET MEMORY CONTROL CLOCK
7000 198 0000002B	CLRMCK	=<^X2B>	;(WO) CLEAR MEMORY CONTROL CLOCK
7000 199			
7000 200 0000002C	CLRTMR	=<^X2C>	;(WO) STOP TIMER
7000 201 0000002D	SETTMR	=<^X2D>	;(WO) START TIMER COUNTING
7000 202			
7000 203 0000002E	CLRBSY	=<^X2E>	;(WO) CLEAR UNIBUS BUS BUSY
7000 204 0000002F	SETBSY	=<^X2F>	;(WO) SET UNIBUS BUS BUSY
7000 205			
7000 206 00000030	SETDCL	=<^X30>	;(WO) SET DC LOW
7000 207 00000031	CLRDCL	=<^X31>	;(WO) CLEAR DC LOW
7000 208			
7000 209 00000032	SETACL	=<^X32>	;(WO) SET AC LOW
7000 210 00000033	CLRACL	=<^X33>	;(WO) CLEAR AC LOW
7000 211			
7000 212 00000034	INITL	=<^X34>	;(WO) CLEAR UNIBUS INIT
7000 213 00000035	INITH	=<^X35>	;(WO) SET UNIBUS INIT
7000 214			
7000 215 00000036	CLRWCK	=<^X36>	;(WO) CLEAR WCS BIT 00
7000 216 00000037	SETWCK	=<^X37>	;(WO) SET WCS BIT 00
7000 217			
7000 218 00000038	CLRCR	=<^X38>	;(WO) CLEAR WCS BIT 01
7000 219 00000039	SETCSR	=<^X39>	;(WO) SET WCS BIT 01
7000 220			
7000 221 0000003C	CLRLIT	=<^X3C>	;(WO) CLEAR THE RUN LIGHT
7000 222 0000003D	SETLIT	=<^X3D>	;(WO) SET THE RUN LIGHT
7000 223			
7000 224 00000044	TUDB	=<^X44>	;(R/W) TU-58 DATA BUFFER
7000 225 00000045	TUSR	=<^X45>	;(RO) TU-58 STATUS REGISTER
7000 226 00000046	TUMODE	=<^X46>	;(R/W) TU-58 MODE REGISTER 1 + 2
7000 227			;(R/W) TU-58 MODE REGISTER 1 + 2
7000 228			;(R/W) TU-58 MODE REGISTER 1 + 2
7000 229			;(R/W) TU-58 MODE REGISTER 1 + 2
7000 230 00000047	TUCR	=<^X47>	;(R/W) TU-58 COMMAND REGISTER
7000 231			
7000 232 00000048	TTDB	=<^X48>	;(R/W) TERMINAL DATA BUFFER
7000 233 00000049	TTSR	=<^X49>	;(RO) TERMINAL STATUS REGISTER
7000 234 0000004A	TTMODE	=<^X4A>	;(R/W) TERMINAL MODE REGISTER 1 + 2
7000 235			;(R/W) TERMINAL MODE REGISTER 1 + 2
7000 236			;(R/W) TERMINAL MODE REGISTER 1 + 2
7000 237			;(R/W) TERMINAL MODE REGISTER 1 + 2
7000 238 0000004B	TTCR	=<^X4B>	;(R/W) TERMINAL COMMAND REGISTER
7000 239			

ZZ-ENKBD-2.0 7000 4

L 1

Fiche 1 Frame L1 Sequence 11

7000 240 00000080	READ	=<^X80>	; (RO) CONSOLE READ REGISTER
7000 241			
7000 242 00000082	CPUACK	=<^X82>	; (RO) ACKNOWLEDGE FROM CPU <LSB>
7000 243			
7000 244 00000083	CPATTN	=<^X83>	; (RO) ATTENTION FROM CPU <LSB>
7000 245			
7000 246 00000084	UPC14	=<^X84>	; (RO) UPC BIT 14 <LSB>
7000 247			
7000 248 00000085	CSR07	=<^X85>	; (RO) CSR BIT 7 <LSB>
7000 249			
7000 250 00000086	CSR15	=<^X86>	; (RO) CSR BIT 15 <LSB>
7000 251			
7000 252 00000087	CSR23	=<^X87>	; (RO) CSR BIT 23 <LSB>
7000 253			
7000 254 000000A0	ENBPAR	=<^XA0>	; (WO) ENABLE STALL ON PARITY ERROR
7000 255 000000A1	DISPAR	=<^XA1>	; (WO) DISABLE STALL ON PARITY ERROR
7000 256			
7000 257 000000A2	VSHIFT	=<^XA2>	; (WO) SET THE V-BUS TO SHIFT MODE
7000 258 000000A3	VNORML	=<^XA3>	; (WO) SET THE V-BUS TO NORMAL MODE
7000 259			
7000 260 000000A4	SETTIM	=<^XA4>	; (WO) SET THE INTERVAL TIMER INTERRUPT
7000 261 000000A5	CLRTIM	=<^XA5>	; (WO) CLEAR THE INTERVAL TIMER INTRPT.
7000 262			
7000 263 000000A6	ENBMEM	=<^XA6>	; (WO) ENABLE MEMORY REFERENCES
7000 264 000000A7	DISMEM	=<^XA7>	; (WO) DISABLE MEMORY REFERENCES
7000 265			
7000 266 000000A8	SETACK	=<^XA8>	; (WO) SET CONSOLE ACKNOWLEDGE
7000 267 000000A9	CLRACK	=<^XA9>	; (WO) CLEAR CONSOLE ACKNOWLEDGE
7000 268			
7000 269			
7000 270			
7000 271			
7000 272			
7000 273 000000A8	CLRUPC	=<^XA8>	; (WO) CLEAR THE V-BUS SERIAL INPUT
7000 274 000000A9	SETUPC	=<^XA9>	; (WO) SET THE V-BUS SERIAL INPUT
7000 275			
7000 276 000000AA	SETATN	=<^XAA>	; (WO) SET CONSOLE ATTENTION
7000 277 000000AB	CLRATN	=<^XAB>	; (WO) CLEAR CONSOLE ATTENTION
7000 278			
7000 279 000000AC	SETPFI	=<^XAC>	; (WO) SET POWER FAIL INTERRUPT
7000 280 000000AD	CLRPF1	=<^XAD>	; (WO) CLEAR POWER FAIL INTERRUPT
7000 281			
7000 282 000000AE	SETHLT	=<^XAE>	; (WO) SET THE CONSOLE HALT BIT
7000 283 000000AF	CLRHLT	=<^XAF>	; (WO) CLEAR THE CONSOLE HALT BIT
7000 284			
7000 285 000000CC	WRITE	=<^XCC>	; (WO) THE CONSOLE WRITE REGISTER
7000 286			
7000 287 000000EC	YBUSRD	=<^XEC>	; (RO) Y-BUS READ
7000 288			
7000 289			
7000 290			
7000 291			
7000 292			
7000 293 000000FF	RESET DATA	= <^XFF>	; EXECUTE A MASTER RESET.
7000 294 00000017	MASTER MODE	= <^X17>	; POINT TO THE MASTER MODE REG.
7000 295 00000000	MODE_REG	= <^X00>	; POINT TO A COUNTER MODE REG.
7000 296 00000008	LOAD_REG	= <^X08>	; POINT TO A COUNTER LOAD REG.
7000 297 00000010	HOLD_REG	= <^X10>	; POINT TO A COUNTER HOLD REG.
7000 298 00000007	ALARM_REG1	= <^X07>	; POINT TO ALARM REG 1.
7000 299 0000000F	ALARM_REG2	= <^X0F>	; POINT TO ALARM REG 2.

;NOTE: THE CONSOLE ACKNOWLEDGE SIGNAL IS ALSO USED AS THE UPC
;SERIAL INPUT. WHEN USED THIS WAY THE SENSE IS INVERTED. TO SIMPLIFY
;MATTERS THE NEXT TWO EQUATES ARE INCLUDED

;INTERVAL TIMER DATA

;DATA TO...

7000	300	00000001	7000
7000	301	00000019	
7000	302	000000E8	
7000	303	000000C0	
7000	304	00000040	
7000	305	000000A0	
7000	306	000000E8	
7000	307	000000E0	
7000	308	000000F0	
7000	309		
7000	310		
7000	311		
7000	312		
7000	313	00004C22	
7000	314	00004C25	
7000	315	000040D9	
7000	316		
7000	317		
7000	318		
7000	319		
7000	320		
7000	321		
7000	322		
7000	323		
7000	324	031A'C3	
7003	325		
7003	326		
7003	327		
7003	328		
7003	329		
7003	330		
7003	331		
7003	332		
7003	333	0200	
7005	334	44 42	
7007	335		
7007	336		
7007	337		
7007	338		
7007	339		
7007	340		
7007	341		
7007	342		
7007	343	00	
7008	344	00	
7009	345	0000	
700B	346	00	
700C	347		
700C	348		
700C	349		
700C	350	000A	
700E	351		
700E	352	00	
700F	353	00	
7010	354	01	
7011	355	FF	
7012	356	FF	
7013	357	FE	
7014	358		
7014	359	000B	

M 1

```
Fiche 1   Frame M1                               Sequence 12
;POINT TO CG1 MODE REG IN ELEM CYC.
;POINT TO CG1 HOLD REG IN HOLD CYC.
;DISABLE DATA POINTER SEQUENCING.
;DISARM A COUNTER.
;LOAD A COUNTER.
;SAVE A COUNTER'S CONTENTS.
;ASSERT A COUNTER'S OUTPUT.
;CLEAR A COUNTER'S OUTPUT.
;SINGLE STEP A COUNTER.
```

Sequence 12

```
CG1_MODE_E           = <^X01>
CG1_HOLD_H           = <^X19>
DIS_DP_SEQ           = <^XE8>
DISARM_CNTR_DAT      = <^XC0>
LOAD_CNTR_DAT        = <^X40>
SAVE_CNTR_DAT        = <^XA0>
SET_OUTPUT_DAT       = <^XE8>
CLR_OUTPUT_DAT       = <^XE0>
INC_CNTR_DAT         = <^XF0>
```

:ADDITIONAL DATA

```

PARITY_VECTOR    = <^X4C22>
POWER_VECTOR     = <^X4C25>
MIPFLG          = <^X40D9>

```

```

;PARITY ERROR BRANCH VECTOR.
;POWER FAIL BRANCH VECTOR.
;ADDRESS OF MODEM IN PROGRESS FLAG

```

```

:*****
:
: PROGRAM ENTRY POINT.... MUST BE THE FIRST INSTUCTION IN THE PROGRAM
:
:*****

```

```
ENTRY:      JMP      TEST1E
```

```

;*****
;
;  VERSION NUMBER AND LAST TWO LETTERS OF FILE NAME.  THESE 2 WORDS CAN NOT
;  BE MOVED TO ANY OTHER LOCATION WITHOUT A MICMON CHANGE.
;*****

```

```

VERSION:      .WORD      <^X0200>      ;VERSION NUMBER (REV 02.00)
FILE_NAME:    .ASCII     'BD'           ;LAST 2 CHARS OF FILE NAME

```

```

*****
:
: VARIABLES AREA
:
*****

```

CG_PNT:	.BYTE	0	:POINTS TO COUNTER GROUP.
SHIFT_CNT:	.BYTE	0	:COUNTS ACCUMULATOR SHIFTS.
TIMER_DAT:	.WORD	0	:DATA TO BE WRITTEN TO TIMER.
CYCLE_CNT:	.BYTE	0	:KEEPS TRACK OF POSITION IN : ELEMENT, HOLD AND CONTROL : GROUP CYCLES.

```
NORMAL_MASTER: .WORD      <^X000A>      ;DATA FOR NORMAL MASTER REG
; EXPECTED BY BOOT CODE
```

```

ZERO_DAT:      .BYTE      <^X00>
ONE_SHIFT:     .BYTE      <^X00>
               .BYTE      <^X01>
ONE_DAT:       .BYTE      <^XFF>
ZERO_SHIFT:    .BYTE      <^XFF>
               .BYTE      <^XFE>

```

TEST1E_DAT: .WORD <^X000B> ;EXPECTED COUNTER MODE DATA

ZZ-ENKBD-2.0 7000 4

N 1

Fiche 1 Frame N1 Sequence 13

7016 360
7016 361
7016 362 5555
7018 363
7018 364
7018 365 000A
701A 366 2800
701C 367
701C 368 0200
701E 369
701E 370 0800
7020 371
7020 372 0C0A
7022 373 0800
7024 374
7024 375
7024 376
7024 377
7024 378
7024 379
7024 380
7024 381
7024 382
7024 383 02
7025 384 00000027
7027 385
7027 386 02
7028 387 0000002A
702A 388
702A 389 02
702B 390 0000002D
702D 391
702D 392
702D 393
702D 394
702D 395
702D 396
702D 397
702D 398
702D 399 40D9 3A
7030 400 B7
7031 401 C8
7032 402 0000'CD
7035 403
7035 404 0000'3A
7038 405 0F
7039 406 0000'DC
703C 407
703C 408 C9
703D 409
703D 410
703D 411
703D 412
703D 413
703D 414
703D 415
703D 416
703D 417
703D 418 FF 3E
703F 419 1D D3

```
TEST22_DAT: .WORD    <^X5555>      ;ALTERNATING PATTERN OF ONES
                                     ; AND ZEROS.

TEST23_DAT: .WORD    <^X000A>      ;MASTER MODE DATA.
                                     ;COUNTER MODE DATA.
               .WORD    <^X2800>

TEST24_DAT: .WORD    <^X0200>      ;COUNTER MODE DATA.

TEST25_DAT: .WORD    <^X0800>      ;COUNTER MODE DATA.

TEST26_DAT: .WORD    <^X0C0A>      ;MASTER MODE REGISTER DATA.
               .WORD    <^X0800>      ;COUNTER MODE REGISTER DATA.
```

```
*****
:
:  SHIFT VARIABLES - DO NOT SEPARATE PAIRS
:
*****
```

```
TEMP_SHIFT: .BYTE    2      ;TEMPORARY STORAGE LOCATION FOR
TEMP_DAT:   .BLKB    2      ; DATA FROM VARIOUS TESTS.

EXP_SHIFT:  .BYTE    2      ;LOCATION OF EXPECTED DATA FROM
EXP_DAT:    .BLKB    2      ; VARIOUS TESTS.

REC_SHIFT:  .BYTE    2      ;LOCATION OF RETURNED DATA FROM
REC_DAT:    .BLKB    2      ; VARIOUS TESTS.
```

```
*****
:
:  CHECK_RD  THIS ROUTINE CHECKS TO SEE IF DIAGNOSTIC IS UNDER RD CONTROL.
:             IF IT IS, A CALL IS MADE TO GCVECT SO THAT THE BOOT CODE CAN
:             KEEP TRACK OF ELAPSED TIME.
:
*****
```

```
CHECK_RD:    LDA      MIPFLG      ;GET STATE OF MODEM IN PROGRESS
               ORA      A          ;SET CONDITION CODES
               RZ              ;RETURN IF ZERO
               CALL     GCVECT     ;ELSE GO TO BOOT CODE IF RD

               LDA      CNTLC_Typed ;SEE IF CONTROL C TYPED (UNDER RD)
               RRC
               CC      SET_FOR_CONT ;IF SO, GO TO MONITOR.

               RET              ;BACK TO TEST
```

```
*****
:
:  MASTER_RESET THIS ROUTINE PERFORMS AN INTERVAL TIMER MASTER RESET.
:
*****
```

```
MASTER_RESET: MVI      A,RESET_DATA ;MOVE MASTER RESET DATA TO A.
               OUT      ITCR        ;PERFORM MASTER RESET.
```

```

ZZ-ENKBD-2.0  7000  4
7041  420
7041  421 C9
7042  422
7042  423
7042  424
7042  425
7042  426
7042  427
7042  428
7042  429
7042  430
7042  431
7042  432
7042  433 FF 3E
7044  434 1D D3
7046  435
7046  436 00'0C'21
7049  437 0057'CD
704C  438
704C  439 C9
704D  440
704D  441
704D  442
704D  443
704D  444
704D  445
704D  446
704D  447
704D  448 AF
704E  449 0007'32
7051  450
7051  451 C9
7052  452
7052  453
7052  454
7052  455
7052  456
7052  457
7052  458
7052  459
7052  460
7052  461 00'07'21
7055  462 34
7056  463
7056  464 C9
7057  465
7057  466
7057  467
7057  468
7057  469
7057  470
7057  471
7057  472
7057  473
7057  474
7057  475 17 3E
7059  476 1D D3
705B  477
705B  478 00D9'CD
705E  479

```

B 2

Fiche 1 Frame B2

Sequence 14

RET

```

*****
: MASTER_RESET_RESTORE THIS ROUTINE PERFORMS AN INTERVAL TIMER MASTER RESET.
: IT THEN WRITES THE MASTER MODE REGISTER TO 0A00 WHICH IS THE
: VALUE EXPECTED TO BE THERE IN THE BOOT BLOCK.
*****

```

MASTER_RESET_RESTORE:

```

MVI A,RESET_DATA ;MOVE MASTER RESET DATA TO A.
OUT ITCR          ;PERFORM MASTER RESET.

LXI H,NORMAL_MASTER ;POINT TO 0A00 DATA
CALL WRITE_MASTER  ;WRITE NORMAL MASTER REG DATA

RET

```

```

*****
: CLR.CG_PNT THIS ROUTINE CLEARS THE COUNTER GROUP POINTER.
*****

```

```

CLR.CG_PNT:  XRA A          ;CLEAR A.
              STA CG_PNT    ;STORE IN CG_PNT.

RET

```

```

*****
: INC.CG_PNT THIS ROUTINE INCREMENTS THE COUNTER GROUP POINTER.
*****

```

```

INC.CG_PNT:  LXI H,CG_PNT   ;POINT TO CG_PNT.
              INR M          ;INCREMENT CG_PNT.

RET

```

```

*****
: WRITE_MASTER THIS ROUTINE WRITE THE DATA, POINTED TO BY THE H,L PAIR,
: TO THE MASTER MODE REGISTER.
*****

```

```

WRITE_MASTER: MVI A,MASTER_MODE ;POINT TO THE MASTER MODE REG.
              OUT ITCR
              CALL WRITE_ITDB    ;WRITE THE DATA TO THE MASTER REG.

```



```

ZZ-ENKBD-2.0  7000  4
705E 480 C9
705F 481
705F 482
705F 483
705F 484
705F 485
705F 486
705F 487
705F 488
705F 489
705F 490
705F 491
705F 492 17 3E
7061 493 1D D3
7063 494
7063 495 00E2'CD
7066 496
7066 497 00'0C'21
7069 498 0057'CD
706C 499
706C 500 C9
706D 501
706D 502
706D 503
706D 504
706D 505
706D 506
706D 507
706D 508
706D 509
706D 510
706D 511 00 06
706F 512
706F 513
706F 514 0007'3A
7072 515 B0
7073 516 1D D3
7075 517
7075 518 00D9'CD
7078 519
7078 520 C9
7079 521
7079 522
7079 523
7079 524
7079 525
7079 526
7079 527
7079 528
7079 529
7079 530
7079 531 00 06
707B 532
707B 533
707B 534 0007'3A
707E 535 B0
707F 536 1D D3
7081 537
7081 538 00E2'CD
7084 539

```

C 2

Fiche 1 Frame C2

Sequence 15

RET

```

*****
: READ_MASTER THIS ROUTINE READS THE MASTER MODE REGISTER AND PUTS THE
: DATA IN REC_DAT. IT THEN REWRITES THE MASTER MODE REG TO
: NORMAL OPERATION (WRITES 0A00 INTO MASTER MODE REG).
*****

```

```

READ_MASTER: MVI A,MASTER_MODE ;POINT TO THE MASTER MODE REG.
              OUT ITCR
              CALL READ_ITDB ;READ THE MASTER MODE REGISTER.
              LXI H,NORMAL_MASTER ;POINT TO 0A00 DATA
              CALL WRITE_MASTER ;WRITE NORMAL MASTER REG DATA
              RET

```

```

*****
: WRITE_MODE THIS ROUTINE WRITES THE DATA, POINTED TO BY THE H,L PAIR,
: TO THE MODE REGISTER POINTED TO BY THE COUNTER GROUP POINTER.
*****

```

```

WRITE_MODE: MVI B,MODE_REG ;DATA FOR MODE REGISTER WITH COUNTER
              ; GROUP FIELD CLEARED.
              LDA CG_PNT ;OR IN THE COUNTER GROUP POINTER.
              ORA B
              OUT ITCR ;POINT TO THE CGX MODE REGISTER.
              CALL WRITE_ITDB ;WRITE THE DATA TO THE MODE REGISTER.
              RET

```

```

*****
: READ_MODE THIS ROUTINE READS THE DATA FROM THE MODE REGISTER POINTED TO
: BY THE COUNTER GROUP POINTER AND PUTS IT IN REC_DAT.
*****

```

```

READ_MODE: MVI B,MODE_REG ;DATA FOR MODE REGISTER WITH COUNTER
              ; GROUP FIELD CLEARED.
              LDA CG_PNT ;OR IN THE COUNTER GROUP POINTER.
              ORA B
              OUT ITCR ;POINT TO THE CGX MODE REGISTER.
              CALL READ_ITDB ;READ THE COUNTER MODE REGISTER.

```

ZZ-ENKBD-2.0 7000 4
7084 540 C9
7085 541
7085 542
7085 543
7085 544
7085 545
7085 546
7085 547
7085 548
7085 549
7085 550
7085 551 08 06
7087 552
7087 553
7087 554 0007'3A
708A 555 B0
708B 556 1D D3
708D 557
708D 558 00D9'CD
7090 559
7090 560 C9
7091 561
7091 562
7091 563
7091 564
7091 565
7091 566
7091 567
7091 568
7091 569
7091 570 08 06
7093 571
7093 572
7093 573 0007'3A
7096 574 B0
7097 575 1D D3
7099 576
7099 577 00E2'CD
709C 578
709C 579 C9
709D 580
709D 581
709D 582
709D 583
709D 584
709D 585
709D 586
709D 587
709D 588
709D 589
709D 590 10 06
709F 591
709F 592
709F 593 0007'3A
70A2 594 B0
70A3 595 1D D3
70A5 596
70A5 597 00D9'CD
70A8 598
70A8 599 C9

D 2
RET

Fiche 1 Frame D2

Sequence 16

```
*****
:
: WRITE_LOAD THIS ROUTINE WRITES THE DATA, POINTED TO BY THE H,L PAIR,
: TO THE LOAD REGISTER POINTED TO BY THE COUNTER GROUP POINTER.
:
*****
```

```
WRITE_LOAD: MVI B,LOAD_REG ;DATA FOR LOAD REGISTER WITH COUNTER
; GROUP FIELD CLEARED.

LDA CG_PNT ;OR IN THE COUNTER GROUP POINTER.
ORA B
OUT ITCR ;POINT TO THE CGX LOAD REGISTER.

CALL WRITE_ITDB ;WRITE THE DATA TO THE LOAD REGISTER.

RET
```

```
*****
:
: READ_LOAD THIS ROUTINE READS THE DATA FROM THE LOAD REGISTER POINTED TO
: BY THE COUNTER GROUP POINTER AND PUTS IT IN REC_DAT.
:
*****
```

```
READ_LOAD: MVI B,LOAD_REG ;DATA FOR LOAD REGISTER WITH COUNTER
; GROUP FIELD CLEARED.

LDA CG_PNT ;OR IN THE COUNTER GROUP POINTER.
ORA B
OUT ITCR ;POINT TO THE CGX LOAD REGISTER.

CALL READ_ITDB ;READ THE DATA FROM THE LOAD REGISTER.

RET
```

```
*****
:
: WRITE_HOLD THIS ROUTINE WRITES THE DATA, POINTED TO BY THE H,L PAIR,
: TO THE HOLD REGISTER POINTED TO BY THE COUNTER GROUP POINTER.
:
*****
```

```
WRITE_HOLD: MVI B,HOLD_REG ;DATA FOR HOLD REGISTER WITH COUNTER
; GROUP FIELD CLEARED.

LDA CG_PNT ;OR IN THE COUNTER GROUP POINTER.
ORA B
OUT ITCR ;POINT TO THE CGX HOLD REGISTER.

CALL WRITE_ITDB ;WRITE THE DATA TO THE HOLD REGISTER.

RET
```


70A9 600
70A9 601
70A9 602
70A9 603
70A9 604
70A9 605
70A9 606
70A9 607
70A9 608
70A9 609
70A9 610 10 06
70AB 611
70AB 612
70AB 613 0007'3A
70AE 614 B0
70AF 615 1D D3
70B1 616
70B1 617 00E2'CD
70B4 618
70B4 619 C9
70B5 620
70B5 621
70B5 622
70B5 623
70B5 624
70B5 625
70B5 626
70B5 627
70B5 628
70B5 629
70B5 630 07 06
70B7 631
70B7 632 0007'3A
70BA 633 3D
70BB 634 00C0'CA
70BE 635
70BE 636 0F 06
70C0 637 78
70C1 638
70C1 639 1D D3
70C3 640
70C3 641 00D9'CD
70C6 642
70C6 643 C9
70C7 644
70C7 645
70C7 646
70C7 647
70C7 648
70C7 649
70C7 650
70C7 651
70C7 652
70C7 653
70C7 654 07 06
70C9 655
70C9 656 0007'3A
70CC 657 3D
70CD 658 00D2'CA
70DD 659

```
*****
: READ_HOLD THIS ROUTINE READS THE DATA FROM THE HOLD REGISTER POINTED TO
: BY THE COUNTER GROUP POINTER AND PUTS IT IN REC_DAT.
*****
```

```
READ_HOLD: MVI B,HOLD_REG ;DATA FOR HOLD REGISTER WITH COUNTER
; GROUP FIELD CLEARED.

LDA CG_PNT ;OR IN THE COUNTER GROUP POINTER.
ORA B
OUT ITCR ;POINT TO THE CGX HOLD REGISTER.

CALL READ_ITDB ;READ THE DATA FROM THE HOLD REGISTER.

RET
```

```
*****
: WRITE_ALARM THIS ROUTINE WRITES THE DATA, POINTED TO BY THE H,L PAIR, TO
: THE ALARM REGISTER POINTED TO BY THE COUNTER GROUP POINTER.
*****
```

```
WRITE_ALARM: MVI B,ALARM_REG1 ;DATA FOR ALARM REGISTER 1.

LDA CG_PNT ;ARE WE POINTING TO COUNTER GROUP 1.
DCR A
JZ 1$ ;SKIP NEXT INSTRUCTION IF SO.

1$: MVI B,ALARM_REG2 ;NO, LOAD DATA FOR ALARM REGISTER 2.
MOV A,B ;MOVE ALARM DATA INTO A.

OUT ITCR ;POINT TO THE CGX ALARM REGISTER.

CALL WRITE_ITDB ;WRITE THE DATA TO THE ALARM REG.

RET
```

```
*****
: READ_ALARM THIS ROUTINE WRITES THE DATA, POINTED TO BY THE H,L PAIR, TO
: THE ALARM REGISTER POINTED TO BY THE COUNTER GROUP POINTER.
*****
```

```
READ_ALARM: MVI B,ALARM_REG1 ;DATA FOR ALARM REGISTER 1.

LDA CG_PNT ;ARE WE POINTING TO COUNTER GROUP 1.
DCR A
JZ 1$ ;SKIP NEXT INSTRUCTION IF SO.
```

```

ZZ-ENKBD-2.0      7000      4
70D0 660 0F 06
70D2 661 78
70D3 662
70D3 663 1D D3
70D5 664
70D5 665 00E2'CD
70D8 666
70D8 667 C9
70D9 668
70D9 669
70D9 670
70D9 671
70D9 672
70D9 673
70D9 674
70D9 675
70D9 676
70D9 677
70D9 678
70D9 679 23
70DA 680 7E
70DB 681 1C D3
70DD 682
70DD 683 2B
70DE 684 7E
70DF 685 1C D3
70E1 686
70E1 687 C9
70E2 688
70E2 689
70E2 690
70E2 691
70E2 692
70E2 693
70E2 694
70E2 695
70E2 696
70E2 697
70E2 698
70E2 699 1C DB
70E4 700 002C'32
70E7 701
70E7 702 1C DB
70E9 703 002B'32
70EC 704
70EC 705 C9
70ED 706
70ED 707
70ED 708
70ED 709
70ED 710
70ED 711
70ED 712
70ED 713
70ED 714
70ED 715
70ED 716
70ED 717 00'25'21
70F0 718 7E
70F1 719 40 F6

```

```

F 2
Fiche 1 Frame F2 Sequence 18
1$: MVI B,ALARM_REG2 ;NO, LOAD DATA FOR ALARM REGISTER 2.
MOV A,B ;MOVE ALARM DATA INTO A.
OUT ITCR ;POINT TO THE CGX ALARM REGISTER.
CALL READ_ITDB ;READ THE ALARM REGISTER.
RET

*****
WRITE_ITDB THIS ROUTINE WRITES TWO BYTES OF DATA POINTED TO BY THE H,L PAIR
TO THE INTERVAL TIMER DATA BUS. THE DATA POINTER REGISTER SHOULD
BE LOADED PRIOR TO USING THIS ROUTINE.
*****

WRITE_ITDB: INX H ;WRITE THE LOW BYTE TO THE
MOV A,M ; INTERVAL TIMER.
OUT ITDB
DCX H ;WRITE THE HIGH BYTE.
MOV A,M
OUT ITDB
RET

*****
READ_ITDB THIS ROUTINE READS TWO BYTES OF DATA FROM THE INTERVAL TIMER DATA
BUS TO REC_DAT. THE DATA POINTER REGISTER SHOULD BE LOADED PRIOR
TO USING THIS ROUTINE.
*****

READ_ITDB: IN ITDB ;READ THE LOW BYTE AND PUT IT IN
STA REC_DAT+1 ; REC_DAT+1.
IN ITDB ;READ THE HIGH BYTE AND PUT IT IN
STA REC_DAT ; REC_DAT.
RET

*****
MM_BITS THIS ROUTINE MOVES TEMP_DAT TO EXP_DAT AND SETS BIT 14 (DISABLE
DATA POINTER INCREMENT) AND CLEARS BIT 13 (8-BIT DATA BUS) OF
EXP_DAT. TEMP_DAT IS UNCHANGED.
*****

MM_BITS: LXI H,TEMP_DAT ;POINT TO TEMP DAT.
MOV A,M ;MOVE HIGH BYTE INTO A.
ORI <^X40> ;SET BIT 14.

```



```

ZZ-ENKBD-2.0      7000      4
70F3 720 DF E6
70F5 721 00'28'21
70F8 722 77
70F9 723
70F9 724 0026'3A
70FC 725 0029'32
70FF 726
70FF 727 C9
7100 728
7100 729
7100 730
7100 731
7100 732
7100 733
7100 734
7100 735
7100 736
7100 737 00'0B'21
7103 738 34
7104 739
7104 740 C9
7105 741
7105 742
7105 743
7105 744
7105 745
7105 746
7105 747
7105 748
7105 749
7105 750 0007'3A
7108 751 0008'32
710B 752
710B 753 AF
710C 754 3F
710D 755
710D 756 17
710E 757
710E 758 00'08'21
7111 759 35
7112 760 010D'C2
7115 761
7115 762 C0 F6
7117 763
7117 764 1D D3
7119 765
7119 766 C9
711A 767
711A 768
711A 769
711A 770
711A 771
711A 772
711A 773
711A 774
711A 775
711A 776
711A 777
711A 778 0007'3A
711D 779 0008'32

```

G 2

Fiche 1 Frame G2

Sequence 19

```

ANI <^XDF> ;CLEAR BIT 13.
LXI H,EXP_DAT
MOV M,A ;MOVE HIGH BYTE BACK TO EXP_DAT.

LDA TEMP_DAT+1 ;MOVE LOW BYTE OF TEMP_DAT
STA EXP_DAT+1 ; INTO LOW BYTE OF EXP_DAT.

RET

```

```

*****
: INC_CYCLE_CNT THIS ROUTINE INCREMENTS THE VALUE IN CYCLE_CNT.
:
*****

```

```

INC_CYCLE_CNT: LXI H,CYCLE_CNT ;POINT TO CYCLE_CNT.
                INR M ;INCREMENT IT.

                RET

```

```

*****
: DISARM_CNTR THIS ROUTINE DISARMS THE COUNTER SPECIFIED BY CG_PNT.
:
*****

```

```

DISARM_CNTR: LDA CG_PNT ;LOAD THE COUNTER GROUP POINTER IN A.
              STA SHIFT_CNT ;STORE A IN SHIFT_CNT.

              XRA A ;CLEAR A.
              CMC ;SET THE CARRY.

1$: RAL ;SHIFT A THROUGH CARRY.

              LXI H,SHIFT_CNT ;POINT TO SHIFT_CNT.
              DCR M ;DECREMENT SHIFT_CNT.
              JNZ 1$ ;IF ZERO, STOP SHIFTING.

              ORI DISARM_CNTR_DAT ;OR IN DISARM COUNTER DATA.

              OUT ITCR ;DISARM COUNTER.

              RET

```

```

*****
: LOAD_CNTR THIS ROUTINE LOADS THE COUNTER OF THE COUNTER GROUP SPECIFIED BY
: CG_PNT FROM THE SOURCE SPECIFIED BY THE COUNTER MODE REGISTER.
:
*****

```

```

LOAD_CNTR: LDA CG_PNT
            STA SHIFT_CNT ;STORE A IN SHIFT_CNT.

```

ZZ-ENKBD-2.0 7000 4

H 2

Fiche 1 Frame H2

Sequence 20

7120 780
7120 781 AF
7121 782 3F
7122 783
7122 784 17
7123 785
7123 786 00'08'21
7126 787 35
7127 788 0122'C2
712A 789
712A 790 40 F6
712C 791
712C 792 1D D3
712E 793
712E 794
712E 795 C9
712F 796
712F 797
712F 798
712F 799
712F 800
712F 801
712F 802
712F 803
712F 804
712F 805
712F 806 0007'3A
7132 807 0008'32
7135 808
7135 809 AF
7136 810 3F
7137 811
7137 812 17
7138 813
7138 814 00'08'21
713B 815 35
713C 816 0137'C2
713F 817
713F 818 A0 F6
7141 819
7141 820 1D D3
7143 821
7143 822
7143 823 C9
7144 824
7144 825
7144 826
7144 827
7144 828
7144 829
7144 830
7144 831
7144 832
7144 833
7144 834 00'07'21
7147 835 E8 3E
7149 836 B6
714A 837
714A 838 1D D3
714C 839

1\$:

```
XRA    A           ;CLEAR A.
CMC                    ;SET THE CARRY.

RAL                    ;SHIFT A THROUGH CARRY.

LXI     H,SHIFT_CNT  ;POINT TO SHIFT CNT.
DCR     M           ;DECREMENT SHIFT CNT.
JNZ     1$          ;IF ZERO, STOP SHIFTING.

ORI     LOAD_CNTR_DAT ;OR IN LOAD COUNTER DATA.

OUT     ITCR        ;LOAD COUNTER FROM REGISTER SPECIFIED
                    ; BY COUNTER MODE REGISTER.

RET
```

```
*****
:
:  SAVE_CNTR  THIS ROUTINE SAVES THE COUNTER SPECIFIED BY CG_PNT IN THE
:              CORRESPONDING HOLD REGISTER.
:
:*****
```

```
SAVE_CNTR:  LDA     CG_PNT      ;LOAD THE COUNTER GROUP POINTER IN A.
             STA     SHIFT_CNT  ;STORE A IN SHIFT_CNT.

             XRA     A           ;CLEAR A.
             CMC                    ;SET THE CARRY.

1$:          RAL                    ;SHIFT A THROUGH CARRY.

             LXI     H,SHIFT_CNT ;POINT TO SHIFT CNT.
             DCR     M           ;DECREMENT SHIFT CNT.
             JNZ     1$          ;IF ZERO, STOP SHIFTING.

             ORI     SAVE_CNTR_DAT ;OR IN SAVE COUNTER DATA.

             OUT     ITCR        ;SAVE COUNTER IN CORRESPONDING
                                ; SAVE REGISTER.

             RET
```

```
*****
:
:  SET_OUTPUT THIS ROUTINE SET THE OUTPUT BIT OF THE COUNTER GROUP SPECIFIED
:              BY CG_PNT.
:
:*****
```

```
SET_OUTPUT: LXI     H,CG_PNT      ;POINT TO THE COUNTER GROUP.
             MVI     A,SET_OUTPUT_DAT ;OR IN THE SET OUTPUT DATA.
             ORA     M

             OUT     ITCR        ;SET THE OUTPUT BIT.
```



```

ZZ-ENKBD-2.0 7000 4
714C 840 C9
714D 841
714D 842
714D 843
714D 844
714D 845
714D 846
714D 847
714D 848
714D 849
714D 850
714D 851 00'07'21
7150 852 E0 3E
7152 853 B6
7153 854
7153 855 1D D3
7155 856
7155 857 C9
7156 858
7156 859
7156 860
7156 861
7156 862
7156 863
7156 864
7156 865
7156 866
7156 867 00'07'21
7159 868 F0 3E
715B 869 B6
715C 870
715C 871 1D D3
715E 872
715E 873 C9
715F 874
715F 875
715F 876
715F 877
715F 878
715F 879
715F 880
715F 881
715F 882
715F 883
715F 884 00'00'11
7162 885 05 0E
7164 886 0000'CD
7167 887
7167 888 C9
7168 889
7168 890
7168 891
7168 892
7168 893
7168 894
7168 895
7168 896
7168 897
7168 898
7168 899 0028'2A

```

I 2

RET

Fiche 1 Frame 12

Sequence 21

```

*****
: CLR_OUTPUT THIS ROUTINE CLEARS THE OUTPUT BIT OF THE COUNTER GROUP
: SPECIFIED BY CG_PNT.
*****

```

```

CLR_OUTPUT:  LXI    H,CG_PNT      ;POINT TO THE COUNTER GROUP.
              MVI    A,CLR_OUTPUT_DAT ;OR IN THE CLEAR OUTPUT DATA.
              ORA    M
              OUT    ITCR        ;SET THE OUTPUT BIT.
              RET

```

```

*****
: INC_CNTR THIS ROUTINE INCREMENTS THE COUNTER SPECIFIED BY CG_PNT.
*****

```

```

INC_CNTR:    LXI    H,CG_PNT      ;POINT TO THE COUNTER GROUP.
              MVI    A,INC_CNTR_DAT ;OR IN THE INCREMENT COUNTER
              ORA    M             ; DATA.
              OUT    ITCR        ;INCREMENT THE COUNTER.
              RET

```

```

*****
: LOAD_MOD_DAT THIS ROUTINE LOADS THE MODULE DATA POINTED TO BY THE H,L
: PAIR INTO CON_ADD_DAT.
*****

```

```

LOAD_MOD_DAT: LXI    D,CON_MOD_DAT ;LOAD THE MODULE DATA POINTED TO BY
              MVI    C,5           ; THE H,L PAIR INTO CON_MOD_DAT.
              CALL   MOVER
              RET

```

```

*****
: TEST1E_ERROR THIS ROUTINE IS USED BY TEST 1E TO DEFINE THE ERROR VARIABLES
: IN PREPARATION FOR USING THE CON_ERROR ROUTINE.
*****

```

```

TEST1E_ERROR: LHLD   EXP_DAT      ;MOVE EXPECTED DATA TO CON_EXP_DAT.

```

```

ZZ-ENKBD-2.0 7000 4
716B 900 0002'22
716E 901
716E 902 002B'2A
7171 903 0002'22
7174 904
7174 905 FF FF 21
7177 906 0000'22
717A 907 02 0E 00'02'21
      0D 23 77 AF
      0180'C2
7186 908
7186 909 03 0E 00'00'21
      0D 23 77 AF
      018C'C2
7192 910 0007'3A
7195 911 0003'32
7198 912
7198 913 00'00'21
719B 914 015F'CD
719E 915
719E 916 0000'CD
71A1 917
71A1 918 C9
71A2 919
71A2 920
71A2 921
71A2 922
71A2 923
71A2 924
71A2 925
71A2 926
71A2 927
71A2 928
71A2 929 01 3E
71A4 930 0000'32
71A7 931
71A7 932 0028'2A
71AA 933 0002'22
71AD 934
71AD 935 002B'2A
71B0 936 0002'22
71B3 937
71B3 938 FF FF 21
71B6 939 0000'22
71B9 940 00 60 21
71BC 941 0002'22
71BF 942
71BF 943 0000'32 3C AF
71C4 944
71C4 945 00'00'21
71C7 946 015F'CD
71CA 947
71CA 948 0000'CD
71CD 949
71CD 950 C9
71CE 951
71CE 952
71CE 953
71CE 954
71CE 955

```

J 2

Fiche 1 Frame J2

Sequence 22

```

SHLD CON_EXP_DAT+2
LHLD REC_DAT ;MOVE RECEIVED DATA TO CON_REC_DAT.
SHLD CON_REC_DAT+2
LXI H,<^XFFFF> ;LAST TWO BYTES ARE OF INTEREST.
SHLD CON_MSK_DAT
ZERO CON_MSK_DAT+2,2

ZERO CON_ADD_DAT,3 ;STORE COUNTER GROUP POINTER

LDA CG_PNT ; IN ADDITIONAL DATA.
STA CON_ADD_DAT+3
LXI H,MWCS_A ;WCS MODULE BEING TESTED.
CALL LOAD_MOD_DAT
CALL CON_ERROR ; PRINT ERROR MESSAGE.
RET

```

```

:*****
: TEST1F_ERROR THIS ROUTINE IS USED BY TEST 1F TO DEFINE THE ERROR VARIABLES
: IN PREPARATION FOR USING THE CON_ERROR ROUTINE.
:*****

```

```

TEST1F_ERROR: MVI A,1 ;ERROR NUMBER 1.
STA CON_ERR_NUM
LHLD EXP_DAT ;MOVE EXPECTED DATA TO CON_EXP_DAT.
SHLD CON_EXP_DAT+2
LHLD REC_DAT ;MOVE RECEIVED DATA TO CON_REC_DAT.
SHLD CON_REC_DAT+2
LXI H,<^XFFFF> ;BITS 0-13 AND 15 ARE OF INTEREST.
SHLD CON_MSK_DAT
LXI H,<^X0080>
SHLD CON_MSK_DAT+2
SET NA_OTHER ;NO ADDITIONAL DATA.
LXI H,MWCS_A ;WCS MODULE BEING TESTED.
CALL LOAD_MOD_DAT
CALL CON_ERROR ; PRINT ERROR MESSAGE.
RET

```

```

:*****
: TEST21_ERROR ROUTINE USED BY TESTS 21,23 AND 25 TO DEFINE THE ERROR

```



```

ZZ-ENKBD-2.0  7000  4
71CE 956
71CE 957
71CE 958
71CE 959
71CE 960
71CE 961 01 3E
71D0 962 0000'32
71D3 963
71D3 964 0028'2A
71D6 965 0C02'22
71D9 966
71D9 967 002B'2A
71DC 968 0002'22
71DF 969
71DF 970 FF FF 21
71E2 971 0000'22
71E5 972 02 0E 00'02'21
      0D 23 77 AF
      01EB'C2
71F1 973
71F1 974 03 0E 00'00'21
      0D 23 77 AF
      01F7'C2
71FD 975 0007'3A
7200 976 0003'32
7203 977
7203 978 00'00'21
7206 979 015F'CD
7209 980
7209 981 0000'CD
720C 982
720C 983 C9
720D 984
720D 985
720D 986
720D 987
720D 988
720D 989
720D 990
720D 991
720D 992
720D 993
720D 994 0028'2A
7210 995 0002'22
7213 996
7213 997 002B'2A
7216 998 0002'22
7219 999
7219 1000 FF FF 21
721C 1001 0000'22
721F 1002 02 0E 00'02'21
      0D 23 77 AF
      0225'C2
722B 1003
722B 1004 03 0E 00'00'21
      0D 23 77 AF
      0231'C2
7237 1005 000B'3A
723A 1006 0003'32
723D 1007

```

K 2

Fiche 1 Frame K2

Sequence 23

VARIABLES IN PREPARATION FOR USING THE CON_ERROR ROUTINE.

```

:
:
:*****
TEST21_ERROR:  MVI    A,1          ;ERROR NUMBER 1.
                STA    CON_ERR_NUM
                LHL    EXP_DAT     ;MOVE EXPECTED DATA TO CON_EXP_DAT.
                SHLD   CON_EXP_DAT+2
                LHL    REC_DAT     ;MOVE RECEIVED DATA TO CON_REC_DAT.
                SHLD   CON_REC_DAT+2
                LXI    H,<^XFFFF> ;LAST TWO BYTES ARE OF INTEREST.
                SHLD   CON_MSK_DAT
                ZERO    CON_MSK_DAT+2,2
                ZERO    CON_ADD_DAT,3 ;ADDITIONAL DATA CONTAINS COUNTER
                LDA    CG_PNT      ; GROUP POINTER.
                STA    CON_ADD_DAT+3
                LXI    H,MWCS_A    ;WCS MODULE BEING TESTED.
                CALL   LOAD_MOD_DAT
                CALL   CON_ERROR    ; PRINT ERROR MESSAGE.
                RET

```

```

:*****
:  TEST22_ERROR THIS ROUTINE IS USED BY TEST 22 TO DEFINE THE ERROR VARIABLES
:  IN PREPARATION FOR USING THE CON_ERROR ROUTINE.
:*****

```

```

TEST22_ERROR:  LHL    EXP_DAT     ;MOVE EXPECTED DATA TO CON_EXP_DAT.
                SHLD   CON_EXP_DAT+2
                LHL    REC_DAT     ;MOVE RECEIVED DATA TO CON_REC_DAT.
                SHLD   CON_REC_DAT+2
                LXI    H,<^XFFFF> ;LAST TWO BYTES ARE OF INTEREST.
                SHLD   CON_MSK_DAT
                ZERO    CON_MSK_DAT+2,2
                ZERO    CON_ADD_DAT,3 ;STORE THE CYCLE COUNTER IN
                LDA    CYCLE_CNT    ; CON_ADD_DAT.
                STA    CON_ADD_DAT+3

```

```

ZZ-ENKBD-2.0 7000 4
723D 1008 00'00'21
7240 1009 015F'CD
7243 1010
7243 1011 0000'CD
7246 1012
7246 1013 C9
7247 1014
7247 1015
7247 1016
7247 1017
7247 1018
7247 1019
7247 1020
7247 1021
7247 1022
7247 1023
7247 1024 0028'3A
724A 1025 0003'32
724D 1026
724D 1027 002B'3A
7250 1028 0003'32
7253 1029
7253 1030 0000'32 3C AF
7258 1031
7258 1032 FF FF 21
725B 1033 0000'22
725E 1034 00 FF 21
7261 1035 0002'22
7264 1036
7264 1037 00'00'21
7267 1038 015F'CD
726A 1039
726A 1040 0000'CD
726D 1041
726D 1042 C9
726E 1043
726E 1044
726E 1045
726E 1046
726E 1047
726E 1048
726E 1049
726E 1050
726E 1051
726E 1052
726E 1053 01 3E
7270 1054 0000'32
7273 1055
7273 1056 0028'3A
7276 1057 0003'32
7279 1058
7279 1059 002B'3A
727C 1060 0003'32
727F 1061
727F 1062 02 0E 00'00'21
0D 23 77 AF
0285'C2
728B 1063 0025'2A
728E 1064 0002'22
7291 1065

```

L 2

Fiche 1 Frame L2 Sequence 24

```

LXI H,MWCS_A ;WCS MODULE BEING TESTED.
CALL LOAD_MOD_DAT
CALL CON_ERROR ; PRINT ERROR MESSAGE.
RET

```

```

*****
: TEST24_ERROR THIS ROUTINE IS USED BY TEST 24 TO DEFINE THE ERROR VARIABLES
: IN PREPARATION FOR USING THE CON_ERROR ROUTINE.
*****

```

```

TEST24_ERROR: LDA EXP_DAT ;MOVE EXPECTED DATA TO CON_EXP_DAT.
STA CON_EXP_DAT+3
LDA REC_DAT ;MOVE RECEIVED DATA TO CON_REC_DAT.
STA CON_REC_DAT+3
SET NA_OTHER ;NO ADDITIONAL DATA.
LXI H,<^XFFFF> ;LAST BYTE IS OF INTEREST.
SHLD CON_MSK_DAT
LXI H,<^X00FF>
SHLD CON_MSK_DAT+2
LXI H,MWCS_A ;WCS MODULE BEING TESTED.
CALL LOAD_MOD_DAT
CALL CON_ERROR ; PRINT ERROR MESSAGE.
RET

```

```

*****
: TEST26_ERROR THIS ROUTINE IS USED BY TEST 26 TO DEFINE THE ERROR VARIABLES
: IN PREPARATION FOR USING THE CON_ERROR ROUTINE.
*****

```

```

TEST26_ERROR: MVI A,1 ;ERROR NUMBER 1.
STA CON_ERR_NUM
LDA EXP_DAT ;MOVE EXPECTED DATA TO CON_EXP_DAT.
STA CON_EXP_DAT+3
LDA REC_DAT ;MOVE RECEIVED DATA TO CON_REC_DAT.
STA CON_REC_DAT+3
ZERO CON_ADD_DAT,2 ;ADDITIONAL DATA CONTAINS DATA SENT
LHLD TEMP_DAT ; TO THE ALARM REGISTER.
SHLD CON_ADD_DAT+2

```



```

ZZ-ENKBD-2.0 7000 4
7291 1066 FF FF 21
7294 1067 0000'22
7297 1068 F9 FF 21
729A 1069 0002'22
729D 1070
729D 1071 00'00'21
72A0 1072 015F'CD
72A3 1073
72A3 1074 0000'CD
72A6 1075
72A6 1076 C9
72A7 1077
72A7 1078
72A7 1079
72A7 1080
72A7 1081
72A7 1082
72A7 1083
72A7 1084
72A7 1085
72A7 1086
72A7 1087
72A7 1088 00'28'21
72AA 1089 006D'CD
72AD 1090
72AD 1091 0079'CD
72B0 1092
72B0 1093 00'28'21
          02 0E 00'2B'11
          02C6'CA 0000'CD
72BE 1094
72BE 1095
72BE 1096 01 3E
72C0 1097 0000'32
72C3 1098
72C3 1099 0168'CD
72C6 1100
72C6 1101
72C6 1102
72C6 1103 0000'3A
72C9 1104 0F
72CA 1105 02A7'DA
72CD 1106
72CD 1107 00'28'21
72D0 1108 0085'CD
72D3 1109
72D3 1110 0091'CD
72D6 1111
72D6 1112 00'28'21
          02 0E 00'2B'11
          02EC'CA 0000'CD
72E4 1113
72E4 1114
72E4 1115 02 3E
72E6 1116 0000'32
72E9 1117
72E9 1118 0168'CD
72EC 1119
72EC 1120
72EC 1121

```

```

*****
*
*
*
*
*
*
*****
*
*
*
*
*
*
*****
*
*
*
*
*
*
*****

```

```

*****
: TEST20_ONES_ZEROS THIS ROUTINE IS USED BY TEST 20 TO CHECK THE MODE,
:                      LOAD, AND HOLD REGISTERS WITH DATA PATTERNS OF ALL
:                      ZEROS OR ALL ONES. EXP_DAT POINTS TO ZEROS OR ONES
:                      DATA.
*****
TEST20_ONES_ZEROS:
LXI H,EXP_DAT ;WRITE EXP_DAT TO THE COUNTER MODE
CALL WRITE_MODE ; REGISTER.
CALL READ_MODE ;READ THE COUNTER MODE REGISTER.
IFNEQ EXP_DAT,REC_DAT,2 ;COMPARE THE EXPECTED AND
                                ; RECEIVED DATA.
MVI A,1 ;ERROR NUMBER 1.
STA CON_ERR_NUM
CALL TEST1E_ERROR ;PRINT AN ERROR MESSAGE.
ENDIF
LDA ERROR_CON ;IF THE ERROR LOOPING FLAG IS SET...
RRC
JC TEST20_ONES_ZEROS ;...JUMP TO BEGINNING OF ERROR LOOP.
1$: LXI H,EXP_DAT ;WRITE EXP_DAT TO THE COUNTER LOAD
CALL WRITE_LOAD ; REGISTER.
CALL READ_LOAD ;READ THE COUNTER LOAD REGISTER.
IFNEQ EXP_DAT,REC_DAT,2 ;COMPARE THE EXPECTED AND
                                ; RECEIVED DATA.
MVI A,2 ;ERROR NUMBER 2.
STA CON_ERR_NUM
CALL TEST1E_ERROR ;PRINT AN ERROR MESSAGE.
ENDIF

```

M 2

Fiche 1 Frame M2 Sequence 25

```

LXI H,<^XFFFF> ;DATA OF INTEREST IS BITS 1 AND 2.
SHLD CON_MSK_DAT
LXI H,<^XF9FF>
SHLD CON_MSK_DAT+2
LXI H,MWCS_A ;WCS MODULE BEING TESTED.
CALL LOAD_MOD_DAT
CALL CON_ERROR ; PRINT ERROR MESSAGE.
RET

```

```

ZZ-ENKBD-2.0 7000 4
72EC 1122 0000'3A
72EF 1123 0F
72F0 1124 02CD'DA
72F3 1125
72F3 1126
72F3 1127 00'28'21
72F6 1128 009D'CD
72F9 1129
72F9 1130 00A9'CD
72FC 1131
72FC 1132 00'28'21
          02 0E 00'28'11
          0312'CA 0000'CD

730A 1133
730A 1134
730A 1135 03 3E
730C 1136 0000'32
730F 1137
730F 1138 0168'CD
7312 1139
7312 1140
7312 1141
7312 1142 0000'3A
7315 1143 0F
7316 1144 02F3'DA
7319 1145
7319 1146 C9
731A 1147
731A 1148
731A 1149
731A 1150
731A 1151
731A 1152
731A 1153
731A 1154
731A 1155
731A 1156
731A 1157
731A 1158
731A 1159
731A 1160
731A 1161
731A 1162
731A 1163
731A 1164
731A 1165
731A 1166
731A 1167
731A 1168
731A 1169
731A 1170
731A 1171
731A 1172
731A 1173
731A 1174
731A 1175
731A 1176
731A 1177
731A 1178
731A 1179

```

```

*****
*
*
*
*
*
*
*****

```

2\$:

```

N 2
LDA ERROR_CON ;IF THE ERROR LOOPING FLAG IS SET...
RRC
JC 1$ ;...JUMP TO BEGINNING OF ERROR LOOP.

LXI H,EXP_DAT ;WRITE EXP_DAT TO THE COUNTER HOLD
CALL WRITE_HOLD ; REGISTER.

CALL READ_HOLD ;READ THE COUNTER HOLD REGISTER.

IFNEQ EXP_DAT,REC_DAT,2 ;COMPARE THE EXPECTED AND
                                ; RECEIVED DATA.

MVI A,3 ;ERROR NUMBER 3.
STA CON_ERR_NUM

CALL TEST1E_ERROR ;PRINT AN ERROR MESSAGE.

ENDIF

LDA ERROR_CON ;IF THE ERROR LOOPING FLAG IS SET...
RRC
JC 2$ ;...JUMP TO BEGINNING OF ERROR LOOP.

RET ;DONE WITH ALL ONES OR ALL ZEROS

```

```

*****
: INTERVAL TIMER TESTS - THE NEXT 14 TESTS DEAL SPECIFICALLY WITH THE
: INTERVAL TIMER CHIP ON THE WCS BOARD.
*****

```

TEST1E

MASTER RESET TEST

```

-----
THIS TEST PERFORMS AN INTERVAL TIMER MASTER RESET BY WRITING
A HEX FF TO THE COMMAND REGISTER. EACH OF THE REGISTERS IS THEN READ
TO SEE THAT IT HAS BEEN PROPERLY INITIALIZED. THE FOLLOWING VALUES
SHOULD BE FOUND IN EACH REGISTER.

```

REGISTER

CONTENTS

```

-----
(5) 16-BIT COUNTER MODE REGISTERS    HEX 0B00
(5) 16 BIT COUNTER LOAD REGISTERS    HEX 0000
(5) 16 BIT COUNTER HOLD REGISTERS    HEX 0000
(1) 16-BIT MASTER MODE REGISTER      HEX 0000

```

TEST STEPS:

ZZ-ENKBD-2.0 7000 4

B 3

Fiche 1 Frame B3

Sequence 27

731A 1180
731A 1181
731A 1182
731A 1183
731A 1184
731A 1185
731A 1186
731A 1187
731A 1188
731A 1189
731A 1190
731A 1191
731A 1192
731A 1193
731A 1194
731A 1195
731A 1196
731A 1197
731A 1198
731A 1199
731A 1200
731A 1201
731A 1202
731A 1203
731A 1204
731A 1205
731A 1206
731A 1207
731A 1208
731A 1209
731A 1210
731A 1211
731A 1212
731A 1213
731A 1214
731A 1215
731A 1216
731A 1217
731A 1218
731A 1219
731A 1220
731A 1221
731A 1222
731A 1223
731A 1224
731A 1225
731A 1226
731A 1227
731A 1228
731A 1229
731A 1230
731A 1231 0000'CD 1E 3E
OF 0000'3A
C9 0327'D2
OF 0000'3A
3D D3 0426'DA
7330 1232
7330 1233 01 3E
7332 1234 0000'32
7335 1235

- 1) PERFORM A MASTER RESET.
- 2) READ EACH REGISTER AND COMPARE EXPECTED AND RECEIVED DATA.
- 3) PRINT ERRORS IF ANY.

ASSUMPTIONS: BECAUSE THIS IS THE FIRST TEST OF THE INTERVAL TIMER CHIP, SEVERAL ASSUMPTIONS MUST BE MADE. IN READING THE CONTENTS OF EACH REGISTER, IT MUST BE ASSUMED THAT THE DATA POINTER IS WORKING CORRECTLY SO THAT THE REGISTER WHICH IS THOUGHT TO BE READ IS THE ONE ACTUALLY BEING READ. ALSO, BECAUSE THE CHIP IS BEING USED IN 8-BIT MODE, THE BYTE POINTER IS ASSUMED TO BE WORKING CORRECTLY, SO THAT THE LOW AND HIGH BYTES OF EACH REGISTER ARE BEING READ CORRECTLY.

ERROR1: MASTER RESET ERROR - MASTER MODE REGISTER.
ERROR2: MASTER RESET ERROR - MODE REGISTER.
ERROR3: MASTER RESET ERROR - LOAD REGISTER.
ERROR4: MASTER RESET ERROR - HOLD REGISTER.

NOTE: ADDITIONAL DATA WILL GIVE COUNTER GROUP FOR ERRORS 2,3 AND 4.

- DEBUG:
- 1) CHECK 'WCSA RESET L' FOR HIGH VALUE DURING MASTER RESET IN MASTER RESET ROUTINE.
 - 2) CHECK 'WCSA SEL TIMER H' FOR HIGH VALUE DURING RESET. IF LOW, CHECK PAL 16L8 (E21).
 - 3) CHECK CHIP SELECT PIN FOR LOW VALUE DURING RESET. IF HIGH CHECK CHIP E17.
 - 4) CHECK 'WCSA A08 H' FOR HIGH VALUE DURING RESET.
 - 5) CHECK 'WCSA WR L' FOR LOW VALUE DURING RESET
 - 6) CHECK 'WCSA RD L' FOR HIGH VALUE DURING RESET.
 - 7) CHECK DB0-DB7 PINS FOR ALL HIGH VALUES MASTER RESET.
 - 8) CHECK X2 PIN FOR 1KHZ SIGNAL.
 - 9) CHECK DB0-DB2 AND DB4 PINS FOR HIGH VALUES DURING WRITE TO COMMAND REGISTER IN READ MASTER ROUTINE.
 - 10) CHECK 'WCSA A08 H' FOR LOW VALUE DURING DATA BUS READ OF READ ITDB ROUTINE.
 - 11) CHECK 'WCSA WR L' FOR HIGH VALUE AND 'WCSA RD L' FOR LOW VALUE DURING THE SAME READ.
 - 12) CHECK DB0-DB7 PINS FOR LOW VALUES DURING THIS READ. IF INCORRECT, OR IF ERRORS OCCUR READING THE CONTENTS OF OTHER REGISTERS, SUSPECT INTERVAL TIMER CHIP.

TEST1E: HEAD <^X1E>

MVI A,1 ;FIRST PART OF TEST 1E.
STA CON_ERR_NUM

ZZ-ENKBD-2.0 7000 4
7414 1343
7414 1344
7414 1345
7414 1346 0000'3A
7417 1347 0F
7418 1348 03FD'DA
741B 1349
741B 1350 35 00'00'21
70 C1 03FA'C2

7424 1351
7424 1352
7424 1353 3C D3
7426 1354
7426 1355
7426 1356
7426 1357
7426 1358
7426 1359
7426 1360
7426 1361
7426 1362
7426 1363
7426 1364
7426 1365
7426 1366
7426 1367
7426 1368
7426 1369
7426 1370
7426 1371
7426 1372
7426 1373
7426 1374
7426 1375
7426 1376
7426 1377
7426 1378
7426 1379
7426 1380
7426 1381
7426 1382
7426 1383
7426 1384
7426 1385
7426 1386
7426 1387
7426 1388
7426 1389
7426 1390
7426 1391
7426 1392
7426 1393
7426 1394
7426 1395
7426 1396
7426 1397
7426 1398
7426 1399 0000'CD 1F 3E
0F 0000'3A
C9 0433'D2

* *
* *****
*
*
*
*
*
*

E 3

Fiche 1 Frame E3

Sequence 30

ENDIF

LDA ERROR_CON ;IF THE ERROR LOOPING FLAG IS SET...
RRC
JC 4\$;...JUMP TO BEGINNING OF ERROR LOOP.

ENDDO

TAIL

TEST1F

MASTER MODE REGISTER TEST

THIS TEST LOADS THE MASTER MODE REGISTER WITH DATA PATTERNS
CONSISTING OF ALL ZEROS, ALL ONES, A ONE SHIFTED THROUGH A ZERO FIELD,
AND A ZERO THROUGH A ONE FIELD. THE DATA SENT IS COMPARED TO THE DATA
RECEIVED TO EVALUATE THE REGISTER'S INTEGRITY. BITS 13 AND 14 ARE
ALWAYS LEFT AT ZERO IN THIS TEST AS THEY CONTROL THE DATA BUS WIDTH
AND DATA POINTER INCREMENTING.

TEST STEPS:

- 1) WRITE HEX FF (MASTER RESET) TO COMMAND REGISTER.
- 2) DISABLE DATA POINTER SEQUENCING.
- 3) WRITE DATA PATTERN TO MASTER MODE REGISTER.
- 4) READ DATA FROM MASTER MODE REGISTER.
- 5) COMPARE SENT AND RECEIVED DATA.
- 6) PRINT ERROR IF ANY.
- 7) REPEAT FOR ALL DATA PATTERNS.

ASSUMPTIONS: THE PREVIOUS INTERVAL TIMER TEST HAS BEEN RUN
SUCCESSFULLY.

ERROR1: MASTER MODE REGISTER ERROR.

DEBUG: SINCE ALL THE EXTERNAL SIGNALS USED PERFORM THIS TEST
WERE USED IN THE PREVIOUS TEST, IF THIS TEST FAILS,
SUSPECT THE INTERVAL TIMER CHIP.

TEST1F: HEAD <^X1F>

ZZ-ENKBD-2.0 7000 4
OF 0000'3A
3D D3 052E'DA

743C 1400
743C 1401 003D'CD
743F 1402
743F 1403 000E'2A
7442 1404 0025'22
7445 1405
7445 1406 00ED'CD
7448 1407
7448 1408 00'28'21
744B 1409 0057'CD
744E 1410
744E 1411 005F'CD
7451 1412
7451 1413 00'28'21
02 0E 00'2B'11
0462'CA 0000'CD

*
*
*
*

745F 1414
745F 1415
745F 1416 01A2'CD
7462 1417
7462 1418
7462 1419
7462 1420 0000'3A
7465 1421 OF
7466 1422 0448'DA
7469 1423
7469 1424
7469 1425
7469 1426 003D'CD
746C 1427
746C 1428 0011'2A
746F 1429 0025'22
7472 1430
7472 1431 00ED'CD
7475 1432
7475 1433
7475 1434 00'28'21
7478 1435 0057'CD
747B 1436
747B 1437 005F'CD
747E 1438
747E 1439 00'28'21
02 0E 00'2B'11
048F'CA 0000'CD

*
*
*
*

748C 1440
748C 1441
748C 1442 01A2'CD
748F 1443
748F 1444
748F 1445
748F 1446 0000'3A
7492 1447 OF
7493 1448 0475'DA
7496 1449
7496 1450
7496 1451 002D'CD
7499 1452
7499 1453 003D'CD

F 3

Fiche 1 Frame F3

Sequence 31

CALL MASTER_RESET ;PERFORM A MASTER RESET.
LHLD ZERO_DAT ;LOAD ALL ZEROS INTO TEMP_DAT.
SHLD TEMP_DAT
CALL MM_BITS ;SET BITS: MM14=1, MM13=0, AND PUT
; DATA IN EXP_DAT.
LXI H,EXP_DAT ;WRITE EXP_DAT TO THE MASTER MODE
CALL WRITE_MASTER ; REGISTER.
CALL READ_MASTER ;READ THE MASTER MODE REGISTER.
IFNEQ EXP_DAT,REC_DAT,2 ;COMPARE EXPECTED AND
; RECEIVED DATA.
CALL TEST1F_ERROR ;PRINT AN ERROR MESSAGE.
ENDIF
LDA ERROR_CON ;IF THE ERROR LOOPING FLAG IS SET...
RRC
JC 1\$;...JUMP TO BEGINNING OF ERROR LOOP.
CALL MASTER_RESET ;PERFORM A MASTER RESET.
LHLD ONE_DAT ;LOAD ALL ONES INTO TEMP_DAT.
SHLD TEMP_DAT
CALL MM_BITS ;SET BITS: MM14=1, MM13=0, AND PUT
; DATA IN EXP_DAT.
LXI H,EXP_DAT ;WRITE EXP_DAT TO THE MASTER MODE
CALL WRITE_MASTER ; REGISTER.
CALL READ_MASTER ;READ THE MASTER MODE REGISTER.
IFNEQ EXP_DAT,REC_DAT,2 ;COMPARE EXPECTED AND
; RECEIVED DATA.
CALL TEST1F_ERROR ;PRINT AN ERROR MESSAGE.
ENDIF
LDA ERROR_CON ;IF THE ERROR LOOPING FLAG IS SET...
RRC
JC 2\$;...JUMP TO BEGINNING OF ERROR LOOP.
CALL CHECK_RD ;CALL BOOT CODE IF UNDER RD CONTROL
CALL MASTER_RESET ;PERFORM A MASTER RESET.

1\$:

2\$:

```

74F5 1497
74F5 1498 00ED'CD
74F8 1499
74F8 1500
74F8 1501 00'28'21
74FB 1502 0057'CD
74FE 1503
74FE 1504 005F'CD
7501 1505
7501 1506 00'28'21
02 0E 00'2B'11
0512'CA 0000'CD

```

1

4\$:

```
IFNEQ EXP_DAT,REC_DAT,2
```

2 :COMPARE EXPECTED AND

Sequence 32

[illegible]

ZZ-ENKBD-2.0 7000 4

```
750F 1507 * *
750F 1508 * *
750F 1509 01A2'CD * *
7512 1510 * *
7512 1511 * *****
7512 1512 *
7512 1513 0000'3A *
7515 1514 0F *
7516 1515 04F8'DA *
7519 1516 *
7519 1517 *
7519 1518 37 *
751A 1519 *
751A 1520 00'24'21 *
751D 1521 0000'22 *
7520 1522 0000'CD *
7523 1523 *
7523 1524 35 00'00'21 *****
70 C1 04F5'C2
```

752C 1525
752C 1526
752C 1527 3C D3

752E 1528
752E 1529
752E 1530
752E 1531
752E 1532
752E 1533
752E 1534
752E 1535
752E 1536
752E 1537
752E 1538
752E 1539
752E 1540
752E 1541
752E 1542
752E 1543
752E 1544
752E 1545
752E 1546
752E 1547
752E 1548
752E 1549
752E 1550
752E 1551
752E 1552
752E 1553
752E 1554
752E 1555
752E 1556
752E 1557
752E 1558
752E 1559
752E 1560
752E 1561
752E 1562
752E 1563
752E 1564
752E 1565

H 3

Fiche 1 Frame H3 Sequence 33
; RECEIVED DATA.

```
CALL TEST1F_ERROR ;PRINT AN ERROR MESSAGE.
ENDIF
LDA ERROR_CON ;IF THE ERROR LOOPING FLAG IS SET...
RRC
JC 4$ ;...JUMP TO BEGINNING OF ERROR LOOP.

STC A ;SET THE CARRY.

LXI H,TEMP_SHIFT ;SHIFT THE ZERO THROUGH THE ONE FIELD.
SHLD SHIFT_PNT
CALL SHIFTER

ENDDO
```

TAIL

TEST20

ELEMENT CYCLE REGISTER TEST

THIS TEST LOADS THE MODE, HOLD AND LOAD REGISTERS IN EACH COUNTER GROUP WITH DATA PATTERNS CONSISTING OF ALL ZEROS, ALL ONES, A ONE SHIFTED THROUGH A ZERO FIELD, AND A ZERO SHIFTED THROUGH A ONE FIELD. THE DATA IS THEN READ BACK AND COMPARED WITH THE DATA SENT.

TEST STEPS:

- 1) PERFORM MASTER RESET.
- 2) POINT TO CG1 MODE REGISTER.
- 3) WRITE DATA PATTERN TO MODE REGISTER.
- 4) READ DATA PATTERN FROM MODE REGISTER.
- 5) COMPARE SENT AND RECEIVED DATA.
- 6) PRINT ERRORS IF ANY.
- 7) REPEAT FOR ALL DATA PATTERNS.
- 8) REPEAT FOR THE OTHER 14 REGISTERS IN ELEMENT CYCLE.

ASSUMPTIONS: PREVIOUS INTERVAL TIMER TESTS HAVE RUN SUCCESSFULLY.

ERROR1: ELEMENT CYCLE REGISTER ERROR - COUNTER MODE REGISTER.
ERROR2: ELEMENT CYCLE REGISTER ERROR - COUNTER LOAD REGISTER.
ERROR3: ELEMENT CYCLE REGISTER ERROR - COUNTER HOLD REGISTER.

NOTE: ADDITIONAL DATA WILL SPECIFY COUNTER GROUP.

DEBUG: SUSPECT INTERVAL TIMER CHIP.

```

752E 1566
752E 1567
752E 1568
752E 1569
752E 1570
752E 1571
752E 1572
752E 1573 0000'CD 20 3E
              OF 0000'3A
              C9 053B'D2
              OF 0000'3A
              3D D3 06AA'DA
7544 1574
7544 1575 0042'CD
7547 1576
7547 1577
7547 1578 004D'CD
754A 1579
754A 1580 C5 46 00'00'21
              77 05 3E
7552 1581
7552 1582 0052'CD
7555 1583
7555 1584 000E'2A
7558 1585 0028'22
755B 1586
755B 1587 02A7'CD
755E 1588
755E 1589 0011'2A
7561 1590 0028'22
7564 1591
7564 1592 02A7'CD
7567 1593
7567 1594
7567 1595 000F'2A
756A 1596 0025'22
756D 1597
756D 1598 C5 46 00'00'21
              77 10 3E
7575 1599
7575 1600 0025'2A
7578 1601 0028'22
757B 1602
757B 1603 00'28'21
757E 1604 006D'CD
7581 1605
7581 1606 0079'CD
7584 1607
7584 1608 00'28'21
              02 0E 00'2B'11
              059A'CA 0000'CD
7592 1609
7592 1610
7592 1611 01 3E
7594 1612 0000'32
7597 1613
7597 1614 0168'CD
759A 1615
759A 1616
759A 1617

```

TEST20:

HEAD <^X20>

7\$:

```

CALL MASTER_RESET_RESTORE ;PERFORM A MASTER RESET AND
                           ; RESTORE MASTER REG TO NORMAL VALUE
CALL CLR.CG_PNT           ;CLEAR THE COUNTER GROUP POINTER.
DO 5
CALL INC.CG_PNT           ;INCREMENT THE COUNTER GROUP POINTER.
LHLD ZERO_DAT             ;LOAD ALL ZEROS INTO EXP_DAT.
SHLD EXP_DAT
CALL TEST20_ONES_ZEROS ;CHECK REGISTERS WITH ALL ZEROS DATA
LHLD ONE_DAT              ;LOAD ALL ONES INTO EXP_DAT.
SHLD EXP_DAT
CALL TEST20_ONES_ZEROS ;CHECK REGISTERS WITH ALL ONES DATA
LHLD ONE_SHIFT           ;LOAD A PATTERN TO SHIFT A ONE THROUGH
SHLD TEMP_DAT             ; A ZERO FIELD INTO TEMP_DAT.
DO 16
LHLD TEMP_DAT            ;MOVE TEMP_DAT TO EXP_DAT.
SHLD EXP_DAT
LXI H,EXP_DAT            ;WRITE EXP_DAT TO THE COUNTER MODE
CALL WRITE_MODE          ; REGISTER.
CALL READ_MODE           ;READ THE COUNTER MODE REGISTER.
IFNEQ EXP_DAT,REC_DAT,2  ;COMPARE THE EXPECTED AND
                           ; RECEIVED DATA.
MVI A,1                  ;ERROR NUMBER 1.
STA CON_ERR_NUM
CALL TEST1E_ERROR        ;PRINT AN ERROR MESSAGE.
ENDIF

```


[illegible]

К 3

Fiche 1 Frame K3

Sequence 36

```

DO      16

LHLD    TEMP_DAT      ;MOVE TEMP_DAT TO EXP_DAT.
SHLD    EXP_DAT

LXI     H,EXP_DAT      ;WRITE EXP_DAT TO THE COUNTER MODE
CALL    WRITE_MODE     ; REGISTER.

CALL    READ_MODE      ;READ THE COUNTER MODE REGISTER.

IFNEQ   EXP_DAT,REC_DAT,2      ;COMPARE THE EXPECTED AND

                                   ; RECEIVED DATA.

MVI     A,1            ;ERROR NUMBER 1.
STA     CON_ERR_NUM

CALL    TEST1E_ERROR    ;PRINT AN ERROR MESSAGE.

ENDIF

LDA     ERROR_CON      ;IF THE ERROR LOOPING FLAG IS SET...
RRC
JC      10$            ;...JUMP TO BEGINNING OF ERROR LOOP.

CALL    CHECK_RD       ;CALL BOOT CODE IF UNDER RD CONTROL

LXI     H,EXP_DAT      ;WRITE EXP_DAT TO THE COUNTER LOAD
CALL    WRITE_LOAD     ; REGISTER.

CALL    READ_LOAD      ;READ THE COUNTER LOAD REGISTER.

IFNEQ   EXP_DAT,REC_DAT,2      ;COMPARE THE EXPECTED AND

                                   ; RECEIVED DATA.

MVI     A,2            ;ERROR NUMBER 2.
STA     CON_ERR_NUM

CALL    TEST1E_ERROR    ;PRINT AN ERROR MESSAGE.

ENDIF

LDA     ERROR_CON      ;IF THE ERROR LOOPING FLAG IS SET...
RRC
JC      11$            ;...JUMP TO BEGINNING OF ERROR LOOP.

LXI     H,EXP_DAT      ;WRITE EXP_DAT TO THE COUNTER HOLD
CALL    WRITE_HOLD     ; REGISTER.

CALL    READ_HOLD      ;READ THE COUNTER HOLD REGISTER.

IFNEQ   EXP_DAT,REC_DAT,2      ;COMPARE THE EXPECTED AND

```


[illegible]

ZZ-ENKBD-2.0 7000 4

M 3

Fiche 1 Frame M3

Sequence 38

76AA 1784
76AA 1785
76AA 1786
76AA 1787
76AA 1788
76AA 1789
76AA 1790
76AA 1791

0000'CD 21 3E
OF 0000'3A
C9 06B7'D2
OF 0000'3A
3D D3 07BD'DA

76C0 1792
76C0 1793 0042'CD
76C3 1794
76C3 1795
76C3 1796 004D'CD
76C6 1797
76C6 1798 C5 46 00'00'21
77 02 3E

76CE 1799
76CE 1800 0052'CD
76D1 1801
76D1 1802 000E'2A
76D4 1803 0028'22
76D7 1804
76D7 1805 00'28'21
76DA 1806 00B5'CD
76DD 1807
76DD 1808 00C7'CD
76E0 1809
76E0 1810 00'28'21
02 0E 00'2B'11
06F1'CA 0000'CD

76EE 1811
76EE 1812
76EE 1813 01CE'CD
76F1 1814
76F1 1815

76F1 1816
76F1 1817 0000'3A
76F4 1818 OF
76F5 1819 06D7'DA

76F8 1820
76F8 1821
76F8 1822 0011'2A
76FB 1823 0028'22

76FE 1824
76FE 1825 00'28'21
7701 1826 00B5'CD

7704 1827
7704 1828 00C7'CD
7707 1829

7707 1830 00'28'21
02 0E 00'2B'11
0718'CA 0000'CD

7715 1831
7715 1832
7715 1833 01CE'CD
7718 1834

DEBUG: SUSPECT INTERVAL TIMER CHIP.

TEST21: HEAD <^X21>

CALL MASTER_RESET_RESTORE ;PERFORM A MASTER RESET AND
; RESTORE MASTER REG TO NORMAL VALUE

CALL CLR.CG_PNT ;CLEAR THE COUNTER GROUP POINTER.

DO 2

CALL INC.CG_PNT ;INCREMENT THE COUNTER GROUP POINTER.

LHLD ZERO_DAT ;MOVE ALL ZEROS TO EXP_DAT.
SHLD EXP_DAT

1\$: LXI H,EXP_DAT ;WRITE ALL ZEROS TO THE ALARM REG.
CALL WRITE_ALARM

CALL READ_ALARM ;READ THE ALARM REGISTER.

IFNEQ EXP_DAT,REC_DAT,2 ;COMPARE THE EXPECTED AND
; RECEIVED DATA.

CALL TEST21_ERROR ;PRINT AN ERROR MESSAGE.

ENDIF

LDA ERROR_CON ;IF THE ERROR LOOPING FLAG IS SET...
RRC
JC 1\$;...JUMP TO BEGINNING OF ERROR LOOP.

LHLD ONE_DAT ;MOVE ALL ONES TO EXP_DAT.
SHLD EXP_DAT

2\$: LXI H,EXP_DAT ;WRITE ALL ONES TO THE ALARM REG.
CALL WRITE_ALARM

CALL READ_ALARM ;READ THE ALARM REGISTER.

IFNEQ EXP_DAT,REC_DAT,2 ;COMPARE THE EXPECTED AND

; RECEIVED DATA.

CALL TEST21_ERROR ;PRINT AN ERROR MESSAGE.

77BD 1944
77BD 1945

★ ★ ★ ★ ★ ★ ★ ★ ★ ★ ★ ★ ★ ★ ★ ★

TAIL

- 1) PERFORM MASTER RESET.
- 2) POINT AT CG1 MODE REGISTER.
- 3) WRITE DATA TO THIS REGISTER.
- 4) SEQUENCE THROUGH CYCLE BY DOING 14 READS.
- 5) COMPARE DATA READ FOR EACH READ WITH DATA SENT (SHOULD NOT BE EQUAL).
- 6) DO LAST READ (SHOULD BE CG1 MODE REG).
- 7) COMPARE WITH ORIGINAL DATA SENT (SHOULD BE EQUAL) AND PRINT ERRORS IF ANY.
- 8) REPEAT FOR HOLD AND CONTROL GROUP CYCLES.
- 9) DISABLE DATA POINTER SEQUENCING.
- 10) READ THEN WRITE A DATA PATTERN (SHOULD BE EQUAL).
- 11) PRINT ERRORS IF ANY.

77BD 1946
77BD 1947
77BD 1948
77BD 1949
77BD 1950
77BD 1951
77BD 1952
77BD 1953
77BD 1954
77BD 1955
77BD 1956
77BD 1957
77BD 1958
77BD 1959
77BD 1960
77BD 1961
77BD 1962
77BD 1963
77BD 1964
77BD 1965
77BD 1966
77BD 1967
77BD 1968
77BD 1969
77BD 1970
77BD 1971
77BD 1972
77BD 1973
77BD 1974
77BD 1975
77BD 1976
77BD 1977
77BD 1978
77BD 1979
77BD 1980
77BD 1981
77BD 1982
77BD 1983
77BD 1984
77BD 1985
77BD 1986 0000'CD 22 3E
OF 0000'3A
C9 07CA'D2
OF 0000'3A
3D D3 09A2'DA

77D3 1987
77D3 1988 01 3E
77D5 1989 0000'32
77D8 1990
77D8 1991 0042'CD
77D8 1992
77D8 1993
77D8 1994 000E'2A
77DE 1995 0028'22
77E1 1996
77E1 1997 02 3E
77E3 1998 000B'32
77E6 1999
77E6 2000 01 3E
77E8 2001 1D D3

ASSUMPTIONS: PREVIOUS INTERVAL TIMER TESTS HAVE RUN SUCCESSFULLY.

ERROR1: ELEMENT CYCLE SEQUENCING MODE ERROR.
ERROR2: HOLD CYCLE SEQUENCING MODE ERROR.
ERROR3: CONTROL GROUP CYCLE SEQUENCING MODE ERROR.

NOTE: FOR ERRORS 1-3, ADDITIONAL DATA WILL CONTAIN A NUMBER
TO INDICATE THE REGISTER THAT FAILED AS FOLLOWS:

ELEMENT CYCLE	HOLD CYCLE	CONTROL CYCLE
-----	-----	-----
1 - CG1 MODE	1 - CG1 HOLD	1 - CG1 ALARM
2 - CG1 LOAD	2 - CG2 HOLD	2 - CG2 ALARM
3 - CG1 HOLD	3 - CG3 HOLD	3 - MAST MODE
4 - CG2 MODE	4 - CG4 HOLD	
5 - CG2 LOAD	5 - CG5 HOLD	
6 - CG2 HOLD		
7 - CG3 MODE		
8 - CG3 LOAD		
9 - CG3 HOLD		
A - CG4 MODE		
B - CG4 LOAD		
C - CG4 HOLD		
D - CG5 MODE		
E - CG5 LOAD		
F - CG5 HOLD		

ERROR4: DATA POINTER ERROR IN NON-SEQUENCING MODE.

DEBUG: SUSPECT INTERVAL TIMER CHIP

TEST22: HEAD <^X22>

1\$:

MVI A,1 ;FIRST PART OF TEST 22.
STA CON_ERR_NUM
CALL MASTER_RESET_RESTORE ;PERFORM A MASTER RESET AND
; RESTORE MASTER REG TO NORMAL VALUE
LHLD ZERO_DAT ;LOAD ZEROS INTO EXP_DAT.
SHLD EXP_DAT
MVI A,2 ;INDICATES SECOND ELEMENT CYCLE
STA CYCLE_CNT ; REGISTER (CG1 LOAD REG).
MVI A,CG1_MODE_E ;POINT TO CG1 MODE REGISTER (ELEMENT
OUT ITCR ; CYCLE).

25:

[illegible]

```

E 4
Fiche 1 Frame E4 Sequence 43
CALL MASTER_RESET_RESTORE ;PERFORM A MASTER RESET AND
; RESTORE MASTER REG TO NORMAL VALUE

LHLD ZERO_DAT ;LOAD ZEROS INTO EXP_DAT.
SHLD EXP_DAT

MVI A,2 ;INDICATES SECOND REGISTER IN
STA CYCLE_CNT ; HOLD CYCLE (CG1 HOLD REG).

MVI A,CG1_HOLD_H ;POINT TO CG1 HOLD REGISTER (HOLD
OUT ITCSR ; CYCLE).

LDA TEST22_DAT ;WRITE TWO BYTES OF ALTERNATING ONES
OUT ITDB ; AND ZEROS TO THE CG1 HOLD REGISTER.
OUT ITDB

DO 4

IN ITDB ;READ LOW BYTE AN ELEMENT CYCLE
STA REC_DAT+1 ; REGISTER (NOT CG1 HOLD REG).

IN ITDB ;READ HIGH BYTE OF SAME REGISTER.
STA REC_DAT

IFNEQ EXP_DAT,REC_DAT,2 ;COMPARE THE EXPECTED AND
; RECEIVED DATA.

CALL TEST22_ERROR ;PRINT AN ERROR MESSAGE.
ENDIF

CALL INC_CYCLE_CNT ;INCREMENT THE CYCLE COUNTER.
ENDDO

SET CYCLE_CNT ;INDICATES FIRST HOLD CYCLE
; REGISTER (CG1 HOLD REG).

LHLD TEST22_DAT ;LOAD PATTERN OF ONES AND ZEROS
SHLD EXP_DAT ; INTO EXP_DAT.

IN ITDB ;READ LOW BYTE OF WHAT SHOULD BE
STA REC_DAT+1 ; CG1 HOLD REGISTER.

IN ITDB ;READ HIGH BYTE OF SAME REGISTER.
STA REC_DAT

IFNEQ EXP_DAT,REC_DAT,2 ;COMPARE THE EXPECTED AND
; RECEIVED DATA.

CALL TEST22_ERROR ;PRINT AN ERROR MESSAGE.
ENDIF

```

[illegible]

```

22-ENKBD-2.0 7000 4
78C5 2110 0000'3A
78C8 2111 0F
78C9 2112 085A'DA
78CC 2113
78CC 2114
78CC 2115
78CC 2116 03 3E
78CE 2117 0000'32
78D1 2118
78D1 2119 002D'CD
78D4 2120
78D4 2121 0042'CD
78D7 2122
78D7 2123
78D7 2124 000E'2A
78DA 2125 0028'22
78DD 2126
78DD 2127 02 3E
78DF 2128 000B'32
78E2 2129
78E2 2130 07 3E
78E4 2131 1D D3
78E6 2132
78E6 2133 0016'3A
78E9 2134 1C D3
78EB 2135 1C D3
78ED 2136
78ED 2137 1C DB
78EF 2138 002C'32
78F2 2139
78F2 2140 1C DB
78F4 2141 002B'32
78F7 2142
78F7 2143 00'28'21
02 0E 00'2B'11
0908'CA 0000'CD

7905 2144
7905 2145
7905 2146 020D'CD
7908 2147
7908 2148
7908 2149
7908 2150 0100'CD
790B 2151
790B 2152
790B 2153 000C'2A
790E 2154 0028'22
7911 2155
7911 2156 1C DB
7913 2157 002C'32
7916 2158
7916 2159 1C DB
7918 2160 002B'32
791B 2161
791B 2162 00'28'21
02 0E 00'2B'11
092C'CA 0000'CD

7929 2163
7929 2164
7929 2165 020D'CD

```

3\$:

```

*****
*
*
*
*
*
*****

```

```

*****
*
*
*
*
*

```

```

F 4
LDA ERROR_CON ;FICHE 1 Frame F4 Sequence 44
RRC ;IF THE ERROR LOOPING FLAG IS SET...
JC 2$ ;...JUMP TO BEGINNING OF ERROR LOOP.

MVI A,3 ;THIRD PART OF TEST 22.
STA CON_ERR_NUM

CALL CHECK_RD ;CALL BOOT CODE IF UNDER RD CONTROL
CALL MASTER_RESET_RESTORE ;PERFORM A MASTER RESET AND
; RESTORE MASTER REG TO NORMAL VALUE

LHLD ZERO_DAT ;LOAD ZEROS INTO EXP_DAT.
SHLD EXP_DAT

MVI A,2 ;INDICATES SECOND CONTROL GROUP
STA CYCLE_CNT ; CYCLE REGISTER (CG2 ALARM REG).

MVI A,ALARM_REG1 ;POINT TO CG1 ALARM REGISTER (CONTROL
OUT ITCSR ; GROUP CYCLE).

LDA TEST22_DAT ;WRITE TWO BYTES OF ALTERNATING ONES
OUT ITDB ; AND ZEROS TO THE CG1 HOLD REGISTER.
OUT ITDB

IN ITDB ;READ LOW BYTE A CONTROL GROUP
STA REC_DAT+1 ; REGISTER (NOT ALARM 1 REG).

IN ITDB ;READ HIGH BYTE OF SAME REGISTER.
STA REC_DAT

IFNEQ EXP_DAT,REC_DAT,2 ;COMPARE THE EXPECTED AND
; RECEIVED DATA.

CALL TEST22_ERROR ;PRINT AN ERROR MESSAGE.
ENDIF

CALL INC_CYCLE_CNT ;INCREMENT THE CYCLE COUNTER.

LHLD NORMAL_MASTER ;GET EXPECTED DATA (MASTER MODE DATA)
SHLD EXP_DAT ;PUT IN EXPECTED DATA

IN ITDB ;READ LOW BYTE A CONTROL GROUP
STA REC_DAT+1 ; REGISTER (NOT ALARM 1 REG).

IN ITDB ;READ HIGH BYTE OF SAME REGISTER.
STA REC_DAT

IFNEQ EXP_DAT,REC_DAT,2 ;COMPARE THE EXPECTED AND
; RECEIVED DATA.

CALL TEST22_ERROR ;PRINT AN ERROR MESSAGE.

```


ZZ-ENKBD-2.0 7000 4

792C 2166 *
792C 2167 *****

792C 2168
792C 2169
792C 2170 000B'32 3C AF

7931 2171
7931 2172
7931 2173 0016'2A
7934 2174 0028'22

7937 2175
7937 2176 1C DB
7939 2177 002C'32

793C 2178
793C 2179 1C DB
793E 2180 002B'32

7941 2181
7941 2182 00'28'21 *****
02 0E 00'2B'11 *
0952'CA 0000'CD *

794F 2183
794F 2184
794F 2185 020D'CD

7952 2186 *****
7952 2187
7952 2188

7952 2189 0000'3A
7955 2190 0F

7956 2191 08D7'DA
7959 2192

7959 2193
7959 2194 04 3E
795B 2195 0000'32

795E 2196
795E 2197 002D'CD
7961 2198

7961 2199 0042'CD
7964 2200

7964 2201
7964 2202 0016'2A
7967 2203 0028'22

796A 2204
796A 2205 01 3E
796C 2206 1D D3

796E 2207
796E 2208 E8 3E
7970 2209 1D D3

7972 2210
7972 2211 0016'3A
7975 2212 1C D3

7977 2213 1C D3
7979 2214
7979 2215 1C DB

797B 2216 002C'32
797E 2217
797E 2218 1C DB

7980 2219 002B'32
7983 2220
7983 2221 00'28'21 *****
02 0E 00'2B'11 *
0999'CA 0000'CD *

G 4

Fiche 1 Frame G4

Sequence 45

ENDIF

SET CYCLE_CNT ;INDICATES FIRST CONTROL GROUP
; CYCLE REGISTER (CG1 ALARM REG).

LHLD TEST22_DAT ;LOAD PATTERN OF ONES AND ZEROS
SHLD EXP_DAT ; INTO EXP_DAT.

IN ITDB ;READ LOW BYTE OF WHAT SHOULD BE
STA REC_DAT+1 ; CG1 ALARM REGISTER.

IN ITDB ;READ HIGH BYTE OF SAME REGISTER.
STA REC_DAT

IFNEQ EXP_DAT,REC_DAT,2 ;COMPARE THE EXPECTED AND
; RECEIVED DATA.

CALL TEST22_ERROR ;PRINT AN ERROR MESSAGE.

ENDIF

LDA ERROR_CON ;IF THE ERROR LOOPING FLAG IS SET...
RRC
JC 3\$;...JUMP TO BEGINNING OF ERROR LOOP.

MVI A,4 ;FOURTH PART OF TEST 22.
STA CON_ERR_NUM

CALL CHECK_RD ;CALL BOOT CODE IF UNDER RD CONTROL

CALL MASTER_RESET_RESTORE ;PERFORM A MASTER RESET AND
; RESTORE MASTER REG TO NORMAL VALUE

LHLD TEST22_DAT ;LOAD PATTERN OF ONES AND ZEROS
SHLD EXP_DAT ; INTO EXP_DAT

MVI A,CG1_MODE_E ;POINT TO CG1 MODE REGISTER (ELEMENT
OUT ITCSR ; CYCLE).

MVI A,DIS_DP_SEQ ;DISABLE DATA POINTER SEQUENCING.
OUT ITCSR

LDA TEST22_DAT ;WRITE TWO BYTES OF ALTERNATING ONES
OUT ITDB ; AND ZEROS TO THE CG1 MODE REGISTER.
OUT ITDB

IN ITDB ;READ THE LOW BYTE OF THE CG1 MODE
STA REC_DAT+1 ; REGISTER.

IN ITDB ;AND THE HIGH BYTE
STA REC_DAT

IFNEQ EXP_DAT,REC_DAT,2 ;COMPARE THE EXPECTED AND

ZZ-ENKBD-2.0 7000 4
7991 2222
7991 2223
7991 2224 0000'32 3C AF
7996 2225
7996 2226 020D'CD
7999 2227
7999 2228
7999 2229
7999 2230 0000'3A
799C 2231 OF
799D 2232 096A'DA
79A0 2233
79A0 2234
79A0 2235 3C D3
79A2 2236
79A2 2237
79A2 2238
79A2 2239
79A2 2240
79A2 2241
79A2 2242
79A2 2243
79A2 2244
79A2 2245
79A2 2246
79A2 2247
79A2 2248
79A2 2249
79A2 2250
79A2 2251
79A2 2252
79A2 2253
79A2 2254
79A2 2255
79A2 2256
79A2 2257
79A2 2258
79A2 2259
79A2 2260
79A2 2261
79A2 2262
79A2 2263
79A2 2264
79A2 2265
79A2 2266
79A2 2267
79A2 2268
79A2 2269
79A2 2270
79A2 2271
79A2 2272
79A2 2273
79A2 2274
79A2 2275
79A2 2276
79A2 2277
79A2 2278
79A2 2279
79A2 2280 0000'CD 23 3E
OF 0000'3A

*
*
*
*
*

H 4

Fiche 1 Frame H4 Sequence 46
; RECEIVED DATA.

SET NA_OTHER ;NO ADDITIONAL DATA.
CALL TEST22_ERROR ;PRINT AN ERROR MESSAGE.
ENDIF
LDA ERROR_CON ;IF THE ERROR LOOPING FLAG IS SET...
RRC
JC 4\$;...JUMP TO BEGINNING OF ERROR LOOP.

TAIL

TEST23 COUNTER LOAD AND SAVE TEST

THIS TEST LOADS EACH OF THE LOAD REGISTERS WITH DATA PATTERNS AND THEN LOADS ALL THE COUNTERS FROM THE LOAD REGISTERS. THE CONTENTS OF THE COUNTERS ARE THEN SAVED IN THE HOLD REGISTERS, AND READ BACK. THE RECEIVED DATA IS COMPARED WITH THE DATA SENT AND ANY APPROPRIATE ERROR MESSAGES ARE PRINTED. THE DATA PATTERNS CONSIST OF ALL ZEROS ALL ONES, AND A ONE SHIFTED THROUGH A ZERO FIELD.

TEST STEPS:

- 1) PERFORM MASTER RESET.
- 2) POINT TO THE CG1 LOAD REGISTER.
- 3) WRITE 16-BIT DATA PATTERN TO LOAD REGISTER.
- 5) LOAD COUNTER 1 FROM LOAD REGISTER 1.
- 6) SAVE COUNTER 1 IN HOLD REGISTER.
- 7) POINT TO CG1 HOLD REGISTER.
- 8) READ HOLD REGISTER 1 AND COMPARE WITH DATA SENT.
- 9) PRINT ERROR OF ANY.
- 9) REPEAT FOR EVERY DATA PATTERN.
- 10) REPEAT FOR COUNTER GROUPS 2,3,4 AND 5.

ASSUMPTIONS: PREVIOUS INTERVAL TIMER TESTS HAVE RUN SUCCESSFULLY.

ERROR1: LOAD/SAVE TEST ERROR. ADDITIONAL DATA WILL SPECIFY COUNTER GROUP.

DEBUG: SUSPECT INTERVAL TIMER CHIP.

TEST23: HEAD <^X23>

ZZ-ENKBD-2.0 7000 4
C9 09AF'D2
0F 0000'3A
3D D3 0AAE'DA

79B8 2281
79B8 2282 003D'CD
79BB 2283
79BB 2284 00'18'21
79BE 2285 0057'CD
79C1 2286
79C1 2287 0G4D'CD
79C4 2288
79C4 2289 C5 46 00'00'21
77 05 3E
79CC 2290
79CC 2291 0052'CD
79CF 2292
79CF 2293 00'1A'21
79D2 2294 006D'CD
79D5 2295
79D5 2296 35 00'00'21
70 C1 09CC'C2
79DE 2297
79DE 2298 004D'CD
79E1 2299
79E1 2300 C5 46 00'00'21
77 05 3E
79E9 2301
79E9 2302 0052'CD
79EC 2303
79EC 2304 000E'2A
79EF 2305 0028'22
79F2 2306
79F2 2307 00'28'21
79F5 2308 0085'CD
79F8 2309
79F8 2310 0105'CD
79FB 2311
79FB 2312 011A'CD
79FE 2313
79FE 2314 012F'CD
7A01 2315
7A01 2316 00A9'CD
7A04 2317
7A04 2318 00'28'21
02 0E 00'2B'11
0A15'CA 0000'CD
7A12 2319
7A12 2320
7A12 2321 01CE'CD
7A15 2322
7A15 2323
7A15 2324
7A15 2325 0000'3A
7A18 2326 0F
7A19 2327 09F2'DA
7A1C 2328
7A1C 2329
7A1C 2330 0011'2A
7A1F 2331 0028'22
7A22 2332

*

*

*

*

*

*

*

*

*

*

*

*

*

*

*

*

*

*

*

*

*

*

*

*

*

*

*

*

*

*

*

*

*

*

*

*

*

*

*

*

*

*

*

*

*

*

*

*

I 4

Fiche 1 Frame 14

Sequence 47

CALL MASTER_RESET ;PERFORM A MASTER RESET.
LXI H,TEST23_DAT ;INITIALIZE THE MASTER MODE
CALL WRITE_MASTER ; REGISTER TO 0A00
CALL CLR.CG_PNT ;CLEAR THE COUNTER GROUP POINTER.
DO 5
CALL INC.CG_PNT ;INCREMENT THE COUNTER GROUP POINTER.
LXI H,TEST23_DAT+2 ;INITIALIZE EACH COUNTER MODE REG.
CALL WRITE_MODE
ENDDO
CALL CLR.CG_PNT ;CLEAR THE COUNTER GROUP POINTER.
DO 5
CALL INC.CG_PNT ;INCREMENT THE COUNTER GROUP POINTER.
LHLD ZERO_DAT ;MOVE ALL ZEROS TO EXP_DAT.
SHLD EXP_DAT
LXI H,EXP_DAT ;WRITE ALL ZEROS TO THE LOAD REG.
CALL WRITE_LOAD
CALL DISARM_CNTR ;DISARM PRESENT COUNTER.
CALL LOAD_CNTR ;LOAD PRESENT COUNTER FROM LOAD REG.
CALL SAVE_CNTR ;SAVE PRESENT COUNTER IN HOLD REGISTER.
CALL READ_HOLD ;READ THE HOLD REGISTER.
IFNEQ EXP_DAT,REC_DAT,2 ;COMPARE THE EXPECTED AND
; RECEIVED DATA.
CALL TEST21_ERROR ;PRINT AN ERROR MESSAGE.
ENDIF
LDA ERROR_CON ;IF THE ERROR LOOPING FLAG IS SET...
RRC
JC 1\$;...JUMP TO BEGINNING OF ERROR LOOP.
LHLD ONE_DAT ;MOVE ALL ONES TO EXP_DAT.
SHLD EXP_DAT

1\$:

ZZ-ENKBD-2.0 7000 4

7A90 2388 * *
7A90 2389 B7 * *
7A91 2390 * *
7A91 2391 00'24'21 * *
7A94 2392 0000'22 * *
7A97 2393 0000'CD * *
7A9A 2394 * *
7A9A 2395 35 00'00'21 * *****
70 C1 0A5A'C2 *
7AA3 2396 *
7AA3 2397 35 00'00'21 *****
70 C1 09E9'C2

7AAC 2398
7AAC 2399 3C D3

7AAE 2400
7AAE 2401
7AAE 2402
7AAE 2403
7AAE 2404
7AAE 2405
7AAE 2406
7AAE 2407
7AAE 2408
7AAE 2409
7AAE 2410
7AAE 2411
7AAE 2412
7AAE 2413
7AAE 2414
7AAE 2415
7AAE 2416
7AAE 2417
7AAE 2418
7AAE 2419
7AAE 2420
7AAE 2421
7AAE 2422
7AAE 2423
7AAE 2424
7AAE 2425
7AAE 2426
7AAE 2427
7AAE 2428
7AAE 2429
7AAE 2430
7AAE 2431
7AAE 2432
7AAE 2433
7AAE 2434
7AAE 2435
7AAE 2436
7AAE 2437
7AAE 2438
7AAE 2439
7AAE 2440
7AAE 2441

0000'CD 24 3E
OF 0000'3A
C9 0ABB'D2
OF 0000'3A
3D D3 0B77'DA

K 4

Fiche 1 Frame K4

Sequence 49

ORA A ;CLEAR THE CARRY.
LXI H,TEMP SHIFT ;SHIFT THE ONE THROUGH THE ZERO FIELD.
SHLD SHIFT_PNT
CALL SHIFTER
ENDDO
ENDDO
TAIL

TEST24

OUTPUT BIT SET/CLEAR TEST

THIS TEST ENSURES THAT THE COUNTER OUTPUTS CAN BE SET AND
CLEARED BY USING THE COMMAND REGISTER COMMAND SET OUTPUT AND CLEAR
OUTPUT. THE COUNTER OUTPUTS ARE READ THROUGH THE STATUS REGISTER.
THIS TEST IS PERFORMED IN THE ACTIVE HIGH DELAYED TOGGLE MODE.

TEST STEPS:

- 1) PERFORM MASTER RESET.
- 2) SET CG MODE REGISTERS TO ACTIVE HIGH DELAYED TOGGLE.
- 3) PERFORM SET OUTPUT BIT COMMAND FOR EACH COUNTER GROUP.
- 4) READ STATUS REGISTER (SHOULD BE HEX FF).
- 5) PRINT ERRORS IF ANY.
- 6) PERFORM CLEAR OUTPUT BIT COMMAND FOR EACH COUNTER GROUP.
- 7) READ STATUS REGISTER (SHOULD BE HEX C1).
- 8) PRINT ERRORS IF ANY.

ASSUMPTIONS: PREVIOUS INTERVAL TIMER TESTS HAVE RUN SUCCESSFULLY.

ERROR1: COUNTER OUTPUT FAILURE, OUTPUT(S) NOT BEING SET.
ERROR2: COUNTER OUTPUT FAILURE, OUTPUT(S) NOT BEING CLEARED.

DEBUG: SUSPECT INTERVAL TIMER CHIP.

TEST24: HEAD <^X24>

```

ZZ-ENKBD-2.0  7000  4
7AC4 2442
7AC4 2443 01 3E
7AC6 2444 0000'32
7AC9 2445
7AC9 2446 0042'CD
7ACC 2447
7ACC 2448
7ACC 2449 004D'CD
7ACF 2450
7ACF 2451 C5 46 00'00'21
              77 05 3E
7AD7 2452
7AD7 2453 0052'CD
7ADA 2454
7ADA 2455 00'1C'21
7ADD 2456 006D'CD
7AE0 2457
7AE0 2458 35 00'00'21
              70 C1 0AD7'C2
7AE9 2459
7AE9 2460 3E 3E
7AEB 2461 0028'32
7AEE 2462
7AEE 2463 004D'CD
7AF1 2464
7AF1 2465 C5 46 00'00'21
              77 05 3E
7AF9 2466
7AF9 2467 0052'CD
7AFC 2468
7AFC 2469 0105'CD
7AFF 2470
7AFF 2471 0144'CD
7B02 2472
7B02 2473 35 00'00'21
              70 C1 0AF9'C2
7B0B 2474
7B0B 2475 1D DB
7B0D 2476 3E E6
7B0F 2477 002B'32
7B12 2478
7B12 2479 00'28'21
              01 0E 00'2B'11
              0B23'CA 0000'CD
7B20 2480
7B20 2481
7B20 2482 0247'CD
7B23 2483
7B23 2484
7B23 2485
7B23 2486 0000'3A
7B26 2487 0F
7B27 2488 0AEE'DA
7B2A 2489
7B2A 2490
7B2A 2491 002D'CD
7B2D 2492
7B2D 2493 02 3F
7B2F 2494 0000'32
7B32 2495

```

```

*****
*
```

```

*
*
*
*
*
*
```

```

*****
```

```
1$:
```

```

*****
*
```

```

*
*
*
*
*
*
```

```

*****
```

```

*****
*
```

```

*
*
*
*
*
```

```

*****
```

```
L 4
```

```
Fiche 1 Frame L4
```

```
Sequence 50
```

```

MVI  A,1          ;FIRST PART OF TEST 8.
STA  CON_ERR_NUM

CALL  MASTER_RESET_RESTORE ;PERFORM A MASTER RESET AND
                                ; RESTORE MASTER REG TO NORMAL VALUE

CALL  CLR.CG_PNT    ;CLEAR THE COUNTER GROUP POINTER.

DO    5

CALL  INC.CG_PNT    ;INCREMENT THE COUNTER GROUP POINTER.

LXI  H,TEST24_DAT  ;SET EACH COUNTER OUTPUT TO DELAYED
CALL  WRITE_MODE    ; TOGGLE.

ENDDO

MVI  A,<^X3E>      ;LOAD DATA WITH ALL OUTPUT BITS
STA  EXP_DAT        ; SET INTO EXP_DAT.

CALL  CLR.CG_PNT    ;CLEAR THE COUNTER GROUP POINTER.

DO    5

CALL  INC.CG_PNT    ;INCREMENT THE COUNTER GROUP POINTER.

CALL  DISARM_CNTR   ;DISARM THE COUNTER.

CALL  SET_OUTPUT     ;SET THE COUNTER'S OUTPUT.

ENDDO

IN    ITCR          ;READ THE STATUS REGISTER.
ANI   <^X3E>        ;MASK OUT BITS 15,14 AND 0.
STA   REC_DAT        ;STORE IN REC_DAT.

IFNEQ EXP_DAT,REC_DAT,1 ;COMPARE THE EXPECTED AND

                                ; RECEIVED DATA.

CALL  TEST24_ERROR   ;PRINT AN ERROR MESSAGE.

ENDIF

LDA   ERROR_CON      ;IF THE ERROR LOOPING FLAG IS SET...
RRC
JC    1$             ;...JUMP TO BEGINNING OF ERROR LOOP.

CALL  CHECK_RD        ;CALL BOOT CODE IF UNDER RD CONTROL

MVI  A,2
STA  CON_ERR_NUM      ;SECOND PART OF TEST 8.

```



```

ZZ-ENKBD-2.0 7000 4
7B32 2496 0028'32 AF
7B36 2497
7B36 2498 004D'CD
7B39 2499
7B39 2500 C5 46 00'00'21
77 05 3E
7B41 2501
7B41 2502 0052'CD
7B44 2503
7B44 2504 0105'CD
7B47 2505
7B47 2506 0144'CD
7B4A 2507
7B4A 2508 014D'CD
7B4D 2509
7B4D 2510 35 00'00'21
70 C1 0B41'C2
7B56 2511
7B56 2512 1D DB
7B58 2513 3E E6
7B5A 2514 002B'32
7B5D 2515
7B5D 2516 00'28'21
01 0E 00'2B'11
0B6E'CA 0000'CD
7B6B 2517
7B6B 2518
7B6B 2519 0247'CD
7B6E 2520
7B6E 2521
7B6E 2522
7B6E 2523 0000'3A
7B71 2524 0F
7B72 2525 0B36'DA
7B75 2526
7B75 2527
7B75 2528 3C D3
7B77 2529
7B77 2530
7B77 2531
7B77 2532
7B77 2533
7B77 2534
7B77 2535
7B77 2536
7B77 2537
7B77 2538
7B77 2539
7B77 2540
7B77 2541
7B77 2542
7B77 2543
7B77 2544
7B77 2545
7B77 2546
7B77 2547
7B77 2548
7B77 2549
7B77 2550
7B77 2551

```

```

*****
*
*
*
*
*
*
*
*
*****
*
*
*
*
*
*****

```

2\$:

```

M 4
CLEAR EXP_DAT ;ALL OUTPUT BITS CLEAR.
CALL CLR.CG_PNT ;CLEAR THE COUNTER GROUP POINTER.
DO 5
CALL INC.CG_PNT ;INCREMENT THE COUNTER GROUP POINTER.
CALL DISARM_CNTR ;DISARM THE COUNTER.
CALL SET_OUTPUT ;SET THE COUNTER'S OUTPUT.
CALL CLR_OUTPUT ;CLEAR THE COUNTER'S OUTPUT.
ENDDO
IN ITCR ;READ THE STATUS REGISTER (ALL OUTPUT
ANI <^X3E> ;MASK OUT BITS 15,14 AND 0.
STA REC_DAT ;BITS SHOULD BE CLEAR).
IFNEQ EXP_DAT,REC_DAT,1 ;COMPARE THE EXPECTED AND
; RECEIVED DATA.
CALL TEST24_ERROR ;PRINT AN ERROR MESSAGE.
ENDIF
LDA ERROR_CON ;IF THE ERROR LOOPING FLAG IS SET...
RRC
JC 2$ ;...JUMP TO BEGINNING OF ERROR LOOP.
TAIL

```

TEST25

COUNTER CARRY TEST

THIS TEST LOADS DATA PATTERNS INTO THE COUNTER REGISTERS THAT WILL CAUSE A CARRY TO THE NEXT HIGHER BIT. THE COUNTER IS THEN INCREMENTED, AND THE RESULT CHECKED TO DETERMINE IF THE COUNTER IS IN FACT COUNTING PROPERLY.

DATA LOADED: 0000000000000000
0000000000000001
0000000000000011
0000000000000111
0000000000001111
0000000000011111
0000000000111111

DATA EXPECTED: 0000000000000001
0000000000000010
0000000000000100
0000000000001000
0000000000010000
000000000100000
000000001000000

7B77	2552	
7B77	2553	
7B77	2554	
7B77	2555	
7B77	2556	
7B77	2557	
7B77	2558	
7B77	2559	
7B77	2560	
7B77	2561	
7B77	2562	
7B77	2563	
7B77	2564	
7B77	2565	
7B77	2566	
7B77	2567	
7B77	2568	
7B77	2569	
7B77	2570	
7B77	2571	
7B77	2572	
7B77	2573	
7B77	2574	
7B77	2575	
7B77	2576	
7B77	2577	
7B77	2578	
7B77	2579	
7B77	2580	
7B77	2581	
7B77	2582	
7B77	2583	
7B77	2584	
7B77	2585	
7B77	2586	
7B77	2587	
7B77	2588	
7B77	2589	
7B77	2590	
7B77	2591	0000'CD 25 3E OF 0000'3A C9 0B84'D2 OF 0000'3A 3D D3 0C23'DA
7B8D	2592	
7B8D	2593	0042'CD
7B90	2594	
7B90	2595	
7B90	2596	004D'CD
7B93	2597	
7B93	2598	C5 46 00'00'21 77 05 3E
7B9B	2599	
7B9B	2600	0052'CD
7B9E	2601	
7B9E	2602	00'1E'21
7BA1	2603	006D'CD
7BA4	2604	
7BA4	2605	35 00'00'21 70 C1 0B9B'C2

Sequence 52

Frame 1	Frame N4	Sequence 32
0000000001111111		0000000010000000
0000000001111111		0000000010000000
0000000011111111		0000000100000000
0000000111111111		0000010000000000
0000011111111111		0000100000000000
0000111111111111		0001000000000000
0000111111111111		0001000000000000
0001111111111111		0010000000000000
0001111111111111		0010000000000000
0011111111111111		0100000000000000
0111111111111111		1000000000000000

- 1) PERFORM MASTER RESET.
- 2) WRITE HEX 0010 TO ALL COUNTER MODE REGISTER TO COUNT UP.
- 3) WRITE A DATA PATTERN TO COUNTER LOAD REGISTER.
- 4) LOAD COUNTER FROM ASSOCIATED LOAD REGISTER.
- 5) INCREMENT COUNTER USING COMMAND REGISTER COUNTER STEP.
- 6) SAVE COUNTER'S CONTENTS IN ASSOCIATED HOLD REGISTER.
- 7) READ THE HOLD REGISTER AND COMPARE DATA WITH EXPECTED DATA.
- 8) PRINT ERRORS IF ANY.
- 9) REPEAT STEPS 3 - 8 FOR EACH DATA PATTERN.
- 10) REPEAT 1-9 FOR EACH COUNTER GROUP.

ERROR1: COUNTER INCREMENT TEST FAILURE. ADDITIONAL DATA WILL SPECIFY COUNTER GROUP THAT FAILED.

DEBUG: SUSPECT INTERVAL TIMER CHIP.

TEST25: HEAD <^X25>

```
CALL    MASTER_RESET_RESTORE      ;PERFORM A MASTER RESET AND
                                     ; RESTORE MASTER REG TO NORMAL VALUE
CALL    CLR.CG_PNT                ;CLEAR THE COUNTER GROUP POINTER.
DO      5
CALL    INC.CG_PNT                ;INCREMENT THE COUNTER GROUP POINTER.
LXI     H,TEST25 DAT              ;SET EACH COUNTER TO COUNT UP.
CALL    WRITE_MODE
ENDDO
```

ZZ-ENKBD-2.0	7000	4
7BAD 2606		
7BAD 2607	004D'CD	
7BB0 2608		
7BB0 2609	C5 46 00'00'21 77 05 3E	
7BB8 2610		
7BB8 2611	0052'CD	
7BBB 2612		
7BBB 2613	000E'2A	
7BBE 2614	0C25'22	
7BC1 2615		
7BC1 2616	C5 46 00'00'21 77 10 3E	
7BC9 2617		
7BC9 2618	0025'2A	
7BCC 2619	0028'22	
7BCF 2620		
7BCF 2621	00'28'21	
7BD2 2622	0000'CD	
7BD5 2623		
7BD5 2624	00'25'21	
7BD8 2625	0085'CD	
7BDB 2626		
7BDB 2627	0105'CD	
7BDE 2628		
7BDE 2629	011A'CD	
7BE1 2630		
7BE1 2631	0156'CD	
7BE4 2632		
7BE4 2633	012F'CD	
7BE7 2634		
7BE7 2635	00A9'CD	
7BEA 2636		
7BEA 2637	00'28'21 02 0E 00'2B'11 0BFB'CA 0000'CD	
7BF8 2638		
7BF8 2639		
7BF8 2640	01CE'CD	
7BFB 2641		
7BFB 2642		
7BFB 2643		
7BFB 2644	0000'3A	
7BFE 2645	0F	
7BFF 2646	0BD5'DA	
7C02 2647		
7C02 2648	37	
7C03 2649		
7C03 2650	00'24'21	
7C06 2651	0000'22	
7C09 2652	0000'CD	
7C0C 2653		
7C0C 2654	002D'CD	
7C0F 2655		
7C0F 2656	35 00'00'21 70 C1 0BC9'C2	
7C18 2657		
7C18 2658	35 00'00'21 70 C1 0BB8'C2	
7C21 2659		

Line	Altitude	Latitude	Longitude	Time	Remarks
7BAD	2606				
7BAD	2607	004D	CD		
7BB0	2608				
7BB0	2609	C5 46	00'00'21		*****
		77 05	3E		*
7BB8	2610				*
7BB8	2611	0052	CD		*
7BBB	2612				*
7BBB	2613	000E	2A		*
7BBE	2614	0G25	22		*
7BC1	2615				*
7BC1	2616	C5 46	00'00'21		* *****
		77 10	3E		* *
7BC9	2617				* *
7BC9	2618	0025	2A		* *
7BCC	2619	0028	22		* *
7BCF	2620				* *
7BCF	2621	00'28	21		* *
7BD2	2622	0000	CD		* *
7BD5	2623				* *
7BD5	2624	00'25	21		* *
7BD8	2625	0085	CD		* *
7BDB	2626				* *
7BDB	2627	0105	CD		* *
7BDE	2628				* *
7BDE	2629	011A	CD		* *
7BE1	2630				* *
7BE1	2631	0156	CD		* *
7BE4	2632				* *
7BE4	2633	012F	CD		* *
7BE7	2634				* *
7BE7	2635	00A9	CD		* *
7BEA	2636				* *
7BEA	2637	00'28	21		* * *****
		02 0E	00'2B'11		* * *
		0BFB	CA 0000'CD		* * *
7BF8	2638				* * *
7BF8	2639				* * *
7BF8	2640	01CE	CD		* * *
7BFB	2641				* * *
7BFB	2642				* * *****
7BFB	2643				* *
7BFB	2644	0000	3A		* *
7BFE	2645	0F			* *
7BFF	2646	0BD5	DA		* *
7C02	2647				* *
7C02	2648	37			* *
7C03	2649				* *
7C03	2650	00'24	21		* *
7C06	2651	0000	22		* *
7C09	2652	0000	CD		* *
7C0C	2653				* *
7C0C	2654	002D	CD		* *
7C0F	2655				* *
7C0F	2656	35 00'00	21		* *****
		70 C1	0BC9'C2		*
7C18	2657				*
7C18	2658	35 00'00	21		*****
		70 C1	0BB8'C2		
7C21	2659				

7BB8	2610		*
7BB8	2611	0052'CD	*
7BBB	2612		*
7BBB	2613	000E'2A	*
7BBE	2614	0C25'22	*
7BC1	2615		*
7BC1	2616	C5 46 00'00'21	* *****
		77 10 3E	* *

7BC9	2617		★ ★
7BC9	2618	0025'2A	★ ★
7BCC	2619	0028'22	★ ★
7BCF	2620		★ ★
7BCF	2621	00'28'21	★ ★
7BD2	2622	0000'CD	★ ★
7BD5	2623		★ ★

7BD5 2624	00'25'21	★ ★	1\$:
7BD8 2625	0085'CD	★ ★	

7BDB	2626		★ ★
7BDB	2627	0105' CD	★ ★
7BDE	2628		★ ★
7BDE	2629	011A' CD	★ ★
79E1	2630		★ ★
7BE1	2631	0156' CD	★ ★
7BE4	2632		★ ★

7BE4	2633	012F'CD	★ ★
7BE7	2634		★ ★
7BE7	2635	00A9'CD	★ ★
7BEA	2636		★ ★
7BEA	2637	00'28'21	★ ★ ★★★★★★★★★★★★★★
		02 0E 00'2B'11	★ ★ ★
		0BFB'CA 0000'CD	★ ★ ★

```

7BF8 2638          * * *
7BF8 2639          * * *
7BF8 2640 01CE'CD  * * *
7BF8 2641          * * *
7BF8 2642          * * *****
7BF8 2643          * *
7BF8 2644 0000'3A  * *

```

7BFE	2645	0F	★ ★
7BFF	2646	0BD5'DA	★ ★
7C02	2647		★ ★
7C02	2648	37	★ ★
7C03	2649		★ ★
7C03	2650	00'24'21	★ ★
7C06	2651	0000'22	★ ★

```

7C08 2651 0000'CD
7C09 2652 0000'CD
7C0C 2653
7C0C 2654 002D'CD
7C0F 2655
7C0F 2656 35 00'00'21
              70 C1 0BC9'C2
7C18 2657

```

```

7C18 2658 35 00'00'21
70 C1 0BB8'C2
7C21 2659

```

B 5

Fiche 1 Frame B5

Sequence 53

```
CALL CLR.CG_PNT      ;CLEAR THE COUNTER GROUP POINTER.
DO 5
```

CALL INC CG_PNT ;INCREMENT THE COUNTER GROUP POINTER.

```
LHLD    ZERO_DAT    ;MOVE THE FIRST DATA PATTERN TO BE
SHLD    TEMP_DAT    ; SENT TO THE COUNTER TO TEMP_DAT.
```

DO 16

```

LHLD    TEMP_DAT      ;MOVE A DATA PATTERN TO EXP_DAT...
SHLD    EXP_DAT

```

```
LXI      H,EXP_DAT      ;...AND INCREMENT IT.
CALL     INC_WORD
```

```
LXI    H,TEMP_DAT      ;WRITE A DATA PATTERN TO THE LOAD
CALL   WRITE_LOAD      ; REGISTER.
```

CALL DISARM_CNTR ;DISARM THE COUNTER.

CALL LOAD_CNTR ;LOAD THE COUNTER FROM THE LOAD REG.

CALL INC_CNTR :INCREMENT THE COUNTER.

CALL SAVE_CNTR ;SAVE THE COUNTER IN THE HOLD REG.

CALL READ_HOLD ;READ THE HOLD REGISTER.

```
IFNEQ EXP_DAT,REC_DAT,2      ;COMPARE THE EXPECTED AND
```

; RECEIVED DATA.

```
CALL TEST21_ERROR ;PRINT AN ERROR MESSAGE.
```

ENDIF

```
LDA    ERROR_CON      ;IF THE ERROR LOOPING FLAG IS SET...
```

```

JC      1$      ;...JUMP TO BEGINNING OF ERROR LOOP.

```

```
STC                                ;SET THE CARRY.
```

```
LXI    H,TEMP_SHIFT      ;MAKE THE NEXT DATA PATTERN TO BE SENT
SHLD   SHIFT_PNT         ; TO THE COUNTER BY SHIFTING A ONE
CALL   SHIFTER           ; INTO THE PRESENT PATTERN.
```

CALL CHECK RD ;CALL BOOT CODE IF UNDER RD CONTROL

ENDDO

ENDO


```

ZZ-ENKBD-2.0 7000 4
7C21 2660 3C D3
7C23 2661
7C23 2662
7C23 2663
7C23 2664
7C23 2665
7C23 2666
7C23 2667
7C23 2668
7C23 2669
7C23 2670
7C23 2671
7C23 2672
7C23 2673
7C23 2674
7C23 2675
7C23 2676
7C23 2677
7C23 2678
7C23 2679
7C23 2680
7C23 2681
7C23 2682
7C23 2683
7C23 2684
7C23 2685
7C23 2686
7C23 2687
7C23 2688
7C23 2689
7C23 2690
7C23 2691
7C23 2692
7C23 2693
7C23 2694
7C23 2695
7C23 2696
7C23 2697
7C23 2698
7C23 2699
7C23 2700
7C23 2701
7C23 2702
7C23 2703
7C23 2704
7C23 2705 0000'CD 26 3E
              0F 0000'3A
              C9 0C30'D2
              0F 0000'3A
              3D D3 0DB0'DA
7C39 2706
7C39 2707 003D'CD
7C3C 2708
7C3C 2709 00'20'21
7C3F 2710 0057'CD
7C42 2711
7C42 2712 004D'CD
7C45 2713
7C45 2714 C5 46 00'00'21
              77 02 3E

```

C 5

TAIL

Fiche 1 Frame C5

Sequence 54

TEST26

COMPARATOR TEST

THIS TEST LOADS THE ALARM REGISTER WITH DATA PATTERNS AND THE COUNTER WITH THE SAME PATTERN DECREMENTED BY ONE. COUNTERS 1 AND 2 COMPARE ENABLE ARE SET AND THE COUNTERS ARE INCREMENTED. THE COUNTER OUTPUTS ARE THEN CHECKED TO MAKE SURE THAT AN ACTIVE HIGH OUTPUT HAS OCCURED. THE DATA PATTERNS CONSIST OF ALL ONES, A ONE SHIFTED THROUGH A ZERO FIELD AND A ZERO SHIFTED THROUGH A ONE FIELD.

TEST STEPS:

- 1) PERFORM MASTER RESET.
- 2) ENABLE COMPARATORS 1 AND 2.
- 3) LOAD ALARM REGISTERS 1 AND 2 WITH A DATA PATTERN.
- 4) LOAD COUNTERS 1 AND 2 WITH DECREMENTED VALUE OF THE DATA PATTERN.
- 5) INCREMENT COUNTERS ONE AND TWO.
- 6) READ THE STATUS REGISTER (SHOULD BE HEX C7).
- 7) PRINT ERROR IF ANY.
- 8) REPEAT STEPS 3 - 7 FOR ALL DATA PATTERNS.

ASSUMPTIONS: PREVIOUS INTERVAL TIMER TESTS HAVE RUN SUCCESSFULLY.

ERROR1: COUNTERS 1 AND/OR 2 OUTPUT BITS NOT SET.

NOTE: ADDITIONAL DATA CONTAINS DATA SENT TO COMPARATOR.

DEBUG: SUSPECT INTERVAL TIMER CHIP.

TEST26:

HEAD <^X26>

```

CALL MASTER_RESET ;PERFORM A MASTER RESET.
LXI H,TEST26_DAT ;SET COUNTERS 1 AND 2 TO COMPARE
CALL WRITE_MASTER ; WITH ALARM REGISTER MODE.
CALL CLR.CG_PNT ;CLEAR THE COUNTER GROUP POINTER.
DO 2

```

*

ZZ-ENKBD-2.0	7000	4
7C4D	2715	
7C4D	2716	0052'CD
7C50	2717	
7C50	2718	00'22'21
7C53	2719	006D'CD
7C56	2720	
7C56	2721	35 00'00'21 70 C1 0C4D'C2
7C5F	2722	
7C5F	2723	06 3E
7C61	2724	0028'32
7C64	2725	
7C64	2726	004D'CD
7C67	2727	
7C67	2728	C5 46 00'00'21 77 02 3E
7C6F	2729	
7C6F	2730	0052'CD
7C72	2731	
7C72	2732	0011'2A
7C75	2733	0025'22
7C78	2734	0009'22
7C7B	2735	
7C7B	2736	00'09'21
7C7E	2737	00B5'CD
7C81	2738	
7C81	2739	00'09'21
7C84	2740	0000'CD
7C87	2741	
7C87	2742	00'09'21
7C8A	2743	0085'CD
7C8D	2744	
7C8D	2745	011A'CD
7C90	2746	
7C90	2747	0156'CD
7C93	2748	
7C93	2749	35 00'00'21 70 C1 0C6F'C2
7C9C	2750	
7C9C	2751	1D DB
7C9E	2752	06 E6
7CA0	2753	002B'32
7CA3	2754	
7CA3	2755	00'28'21 01 0E 00'2B'11 0CB4'CA 0000'CD
7CB1	2756	
7CB1	2757	
7CB1	2758	026E'CD
7CB4	2759	
7CB4	2760	
7CB4	2761	
7CB4	2762	0000'3A
7CB7	2763	0F
7CB8	2764	0C64'DA
7CBB	2765	
7CBB	2766	
7CBB	2767	000F'2A
7CBE	2768	0025'22
7CC1	2769	

D 5

Fiche 1 Frame D5

Sequence 55

```

CALL      INC_CG_PNT      ;INCREMENT THE COUNTER GROUP POINTER.
LXI       H,TEST26_DAT+2 ;INITIALIZE THE COUNTER MODE REG.
CALL      WRITE_MODE
ENDDO

MVI       A,<^X06>        ;LOAD DATA WITH COUNTER OUPUT BITS 1
STA       EXP_DAT         ; AND 2 SET INTO EXP_DAT.

CALL      CLR_CG_PNT      ;CLEAR THE COUNTER GROUP POINTER.
DO        2

CALL      INC_CG_PNT      ;INCREMENT THE COUNTER GROUP POINTER.
LHLD     ONE_DAT          ;LOAD ALL ONES INTO TEMP_DAT AND
SHLD     TEMP_DAT         ; TIMER_DAT.
SHLD     TIMER_DAT

LXI       H,TIMER_DAT     ;WRITE ALL ONES TO THE ALARM REG.
CALL      WRITE_ALARM

LXI       H,TIMER_DAT     ;DECREMENT THE DATA PATTERN...
CALL      DECR_WORD

LXI       H,TIMER_DAT     ;...AND WRITE IT TO THE LOAD REG.
CALL      WRITE_LOAD

CALL      LOAD_CNTR       ;LOAD THE COUNTER FROM THE LOAD REG.
CALL      INC_CNTR        ;INCREMENT THE COUNTER.
ENDDO

IN        ITCSR           ;READ THE STATUS REG.
ANI       <^X06>          ;MASK OUT BITS 1 AND 3-7.
STA       REC_DAT         ;STORE IN REC_DAT.

IFNEQ     EXP_DAT,REC_DAT,1      ;COMPARE THE EXPECTED AND
                                ; RECEIVED DATA.
CALL      TEST26_ERROR        ;PRINT AN ERROR MESSAGE.
ENDIF

LDA       ERROR_CON        ;IF THE ERROR LOOPING FLAG IS SET...
RRC
JC        1$               ;....JUMP TO BEGINNING OF ERROR LOOP.

LHLD     ONE_SHIFT         ;LOAD PATTERN OF ONE SHIFTED THROUGH
SHLD     TEMP_DAT          ; ZERO FIELD INTO TEMP_DAT.

```


ZZ-ENKBD-2.0 7000 4

G 5

Fiche 1 Frame G5

Sequence 58

7DB0 2878
7DB0 2879
7DB0 2880
7DB0 2881
7DB0 2882
7DB0 2883
7DB0 2884
7DB0 2885 0000'CD 51 3E
OF 0000'3A
C9 0DBD'D2
OF 0000'3A
3D D3 0DC8'DA

7DC6 2886
7DC6 2887 3C D3
7DC8 2888

: LAST TEST FOR EVERY CONSOL_SECTION....FORCES END OF SECTION
: *****

END_TEST: HEAD <^X51>

TAIL

\$PSW	=	00000006		
\$SP	=	00000006		
A	=	00000007		
ALARM_REG1	=	0C000007		
ALARM_REG2	=	0000000F		
ALLOWP	=	00000003		
APTF LG	=	00000004		
AUTO	=	00000005		
B	=	00000000		
BOOTEN	=	00000007		
BOOTF	=	00000001		
C	=	00000001		
CG1_HOLD_H	=	00000019		
CG1_MODE_E	=	00000001		
CG_PNT	=	00000007	R	02
CHECK_RD	=	0000002D	R	02
CLRACK	=	000000A9		
CLRACL	=	00000033		
CLRATN	=	000000AB		
CLRBSY	=	0000002E		
CLRCLK	=	00000020		
CLRCSK	=	00000024		
CLRCSR	=	00000038		
CLRDCL	=	00000031		
CLRHLT	=	000000AF		
CLRLIT	=	0000003C		
CLRMCI	=	00000029		
CLRMCK	=	0000002B		
CLRPFI	=	000000AD		
CLRSS	=	00000022		
CLRTIM	=	000000A5		
CLRTMR	=	0000002C		
CLRUPC	=	000000A8		
CLRUPK	=	00000026		
CLRWCK	=	00000036		
CLR.CG_PNT	=	0000004D	R	02
CLR_OUTPUT	=	0000014D	R	02
CLR_OUTPUT_DAT	=	000000E0		
CNT	=	0000082D		
CNTA	=	00000FA0		
CNTLC_TYPED	=	*****	X	02
COMPARE	=	*****	X	02
CON_ADD_DAT	=	*****	X	02
CON_BEGIN_TEST	=	*****	X	02
CON_ERROR	=	*****	X	02
CON_ERR_NUM	=	*****	X	02
CON_EXP_DAT	=	*****	X	02
CON_MOD_DAT	=	*****	X	02
CON_MSK_DAT	=	*****	X	02
CON_REC_DAT	=	*****	X	02
CPATTN	=	00000083		
CPUACK	=	00000082		
CSR07	=	00000085		
CSR15	=	00000086		
CSR23	=	00000087		
CYCLE_CNT	=	0000000B	R	02
D	=	00000002		

DCLO	=	00000002		
DECR_WORD	=	*****	X	02
DISARM_CNTR	=	00000105	R	02
DISARM_CNTR_DAT	=	000000C0		
DISMEM	=	000000A7		
DISPAR	=	000000A1		
DIS_DP_SEQ	=	000000E8		
DOLoop	=	*****	X	02
E	=	00000003		
ENBMEM	=	000000A6		
ENBPAT	=	000000A0		
END TEST	=	00000DB0	R	02
ENTRY	=	00000000	R	02
EOS_FLG	=	*****	X	02
ERROR_CON	=	*****	X	02
EXP_DAT	=	00000028	R	02
EXP_SHIFT	=	00000027	R	02
FILE_NAME	=	00000005	R	02
GCVECT	=	*****	X	02
H	=	00000004		
HEAD	=	00000000		
HLTFLG	=	00000002		
HOLD_REG	=	00000010		
INC.CG_PNT	=	00000052	R	02
INC_CNTR	=	00000156	R	02
INC_CNTR_DAT	=	000000F0		
INC_CYCLE_CNT	=	00000100	R	02
INC_WORD	=	*****	X	02
INITH	=	00000035		
INITL	=	00000034		
ITCSR	=	0000001D		
ITDB	=	0000001C		
L	=	00000005		
LITERAL	=	00000001		
LOAD_CNTR	=	0000011A	R	02
LOAD_CNTR_DAT	=	00000040		
LOAD_MOD_DAT	=	0000015F	R	02
LOAD_REG	=	00000008		
LVL	=	00000000		
M	=	00000006		
MACSTK1	=	0000082D		
MACSTK2	=	00000829		
MACSTK3	=	0000082B		
MACSTK4	=	00000820		
MASTER_MODE	=	00000017		
MASTER_RESET	=	0000003D	R	02
MASTER_RESET_RESTORE	=	00000042	R	02
MIPFLG	=	000040D9		
MISC2	=	00000001		
MM BITS	=	000000ED	R	02
MODE REG	=	00000000		
MOVER	=	*****	X	02
MWCS_A	=	*****	X	02
NA_OTHER	=	*****	X	02
NORMAL_MASTER	=	0000000C	R	02
OKV15	=	00000007		
ONE_DAT	=	00000011	R	02

ONE_SHIFT	0000000F	R	02
PARITY_VECTOR	= 00004C22		
POWER_VECTOR	= 00004C25		
READ	= 00000080		
READJ1	= 00000003		
READJ2	= 00000004		
READ_ALARM	000000C7	R	02
READ_HOLD	000000A9	R	02
READ_ITDB	000000E2	R	02
READ_LOAD	00000091	R	02
READ_MASTER	0000005F	R	02
READ_MODE	00000079	R	02
REC_DAT	0000002B	R	02
REC_SHIFT	0000002A	R	02
REMDIS	= 00000006		
RESET_DATA	= 000000FF		
SAVE_CNTR	0000012F	R	02
SAVE_CNTR_DAT	= 000000A0		
SETACK	= 000000A8		
SETACL	= 00000032		
SETATN	= 000000AA		
SETBSY	= 0000002F		
SETCLK	= 00000021		
SETCSK	= 00000025		
SETCSR	= 00000039		
SETDCL	= 00000030		
SEHLT	= 000000AE		
SETLIT	= 0000003D		
SETMCI	= 00000028		
SETMCK	= 0000002A		
SETPFI	= 000000AC		
SETSS	= 00000023		
SETTIM	= 000000A4		
SETTMR	= 0000002D		
SETUPC	= 000000A9		
SETUPK	= 00000027		
SETWCK	= 00000037		
SET_FOR_CONT	*****	X	02
SET_OUTPUT	00000144	R	02
SET_OUTPUT_DAT	= 000000E8		
SHIFTER	*****	X	02
SHIFT_CNT	00000008	R	02
SHIFT_PNT	*****	X	02
SKIP_TEST	*****	X	02
SYM	= 0000082D		
TEMP_DAT	00000025	R	02
TEMP_SHIFT	00000024	R	02
TEST1E	0000031A	R	02
TEST1E_DAT	00000014	R	02
TEST1E_ERROR	00000168	R	02
TEST1F	00000426	R	02
TEST1F_ERROR	000001A2	R	02
TEST20	0000052E	R	02
TEST20_ONES_ZEROS	000002A7	R	02
TEST21	000006AA	R	02
TEST21_ERROR	000001CE	R	02
TEST22	000007BD	R	02

TEST22_DAT	00000016	R	02
TEST22_ERROR	0000020D	R	02
TEST23	000009A2	R	02
TEST23_DAT	00000018	R	02
TEST24	00000AAE	R	02
TEST24_DAT	0000001C	R	02
TEST24_ERROR	00000247	R	02
TEST25	00000B77	R	02
TEST25_DAT	0000001E	R	02
TEST26	00000C23	R	02
TEST26_DAT	00000020	R	02
TEST26_ERROR	0000026E	R	02
TIMER_DAT	00000009	R	02
TTCR	= 0000004B		
TTDB	= 00000048		
TTMODE	= 0000004A		
TTSR	= 00000049		
TUCR	= 00000047		
TUDB	= 00000044		
TUMODE	= 00000046		
TUSR	= 00000045		
UPC14	= 00000084		
UPC14A	= 00000000		
VERSION	00000003	R	02
VNORML	= 000000A3		
VSHIFT	= 000000A2		
WCSB0	= 00000008		
WCSB1	= 00000009		
WCSB2	= 0000000A		
WRITE	= 000000CC		
WRITE_ALARM	000000B5	R	02
WRITE_HOLD	0000009D	R	02
WRITE_ITDB	000000D9	R	02
WRITE_LOAD	00000085	R	02
WRITE_MASTER	00000057	R	02
WRITE_MODE	0000006D	R	02
Y	= 00000044		
YBUSRD	= 000000EC		
ZERO_DAT	0000000E	R	02
ZERO_SHIFT	00000012	R	02

+-----+
! Psect synopsis !
+-----+

PSECT name	Allocation	PSECT No.	Attributes
. ABS .	00000000 (0.)	00 (0.)	NOPIC USR CON ABS LCL NOSHR NOEXE NORD NOWRT NOVEC BYTE
. BLANK .	00000000 (0.)	01 (1.)	NOPIC USR CON REL LCL NOSHR EXE RD WRT NOVEC BYTE
CPU4	00000DC8 (3528.)	02 (2.)	NOPIC USR CON REL LCL NOSHR EXE RD WRT NOVEC BYTE

+-----+
! Performance indicators !
+-----+

Phase	Page faults	CPU Time	Elapsed Time
Initialization	23	00:00:00.05	00:00:00.28
Command processing	34	00:00:00.26	00:00:00.76
Pass 1	1355	00:00:32.44	00:00:34.41
Symbol table sort	0	00:00:00.30	00:00:00.33
Pass 2	1409	00:00:12.70	00:00:14.70
Symbol table output	29	00:00:00.16	00:00:00.18
Psect synopsis output	5	00:00:00.02	00:00:00.02
Cross-reference output	0	00:00:00.00	00:00:00.00
Assembler run totals	2859	00:00:45.93	00:00:50.68

The working set limit was 1000 pages.
375846 bytes (735 pages) of virtual memory were used to buffer the intermediate code.
There were 20 pages of symbol table space allocated to hold 211 non-local and 131 local symbols.
4912 source lines were read in Pass 1, producing 44 object records in Pass 2.
151 pages of virtual memory were used to define 150 macros.

+-----+
! Macro library statistics !
+-----+

Macro library name	Macros defined
SYS\$SYSROOT:[SYSLIB]STARLET.MLB;1	0

0 GETS were required to define 0 macros.

There were no errors, warnings or information messages.

/SHOW=(BIN)/LIST=ENKBD.LIS/OBJECT=ENKBD.OBJ 8085MAC.MAR+MACDEF.MAC+[]ENKBD.MAC